TUGAS AKHIR – TE 141599

PENGEMBANGAN *SOFTWARE DYNAMIC ECONOMIC DISPATCH* DENGAN MEMPERHITUNGKAN RUGI-RUGI TRANSMISI MENGGUNAKAN METODE *ENHANCED* LAMBDA ITERASI

Nadia Dwiatmo
NRP. 2213106039

Dosen Pembimbing
Prof. Ir. H. Ontoseno Penangsang, M.Sc, Ph.D
Dedet Candra Riawan, ST., M.Eng., Ph.D

JURUSAN TEKNIK ELEKTRO
Fakultas Teknik Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

THE DEVELOPMENT OF DYNAMIC ECONOMIC DISPATCH SOFTWARE WITH CONSIDERING TRANSMISSION LOSSES USING ENHANCED LAMBDA ITERATION

Nadia Dwiatmo
NRP. 2213106039

Advisors
Prof. Ir. H. Ontoseno Penangsang, M.Sc, Ph.D
Dedet Candra Riawan, ST., M.Eng., Ph.D

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

PENGEMBANGAN *SOFTWARE DYNAMIC ECONOMIC DISPATCH* DENGAN MEMPERHITUNGKAN RUGI-RUGI TRANSMISI MENGGUNAKAN METODE *ENHANCED* LAMBDA ITERASI

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Tenaga
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D	Dedet Candra Riawan, ST., M.Eng., Ph.D
NIP. 19490715 1974121 001	NIP : 197311192 000031 0011

27-01-16

SURABAYA
JANUARI, 2016

Pengambangan *Software Dynamic Economic Dispatch* Dengan Memperhitungkan Rugi-Rugi Transmisi Menggunakan Metode *Enhanced Lambda* Iterasi

Nama : Nadia Dwiatmo
Pembimbing 1 : Prof. Ir. H. Ontoseno Penangsang, M.Sc, Ph.D
Pembimbing 2 : Dedet Candra Riawan, ST., M.Eng., Ph.D

ABSTRAK

Biaya pembangkitan yang ekonomis sangat didukung dengan pengoptimalan sistem yang baik. *Economic Dispatch* adalah ilmu yang khusus mempelajari pengoptimalan sistem tenaga listrik dari sisi pembangkitan daya generator. Sedangkan perubahan permintaan nilai beban dalam suatu waktu interval tertentu akan dipelajari dalam analisis *Dynamic Economic Dispatch*. Diharapkan dengan analisis ini akan didapatkan nilai pembangkitan yang optimal ekonomi. Banyak metode yang digunakan untuk mendapatkan nilai optimal secara ekonomi salah satunya metode *Enhanced Lambda* Iterasi. *Enhanced Lambda* Iterasi merupakan pengembangan dari metode lambda Iterasi. Yang mana diharapkan akan mendapatkan nilai pembangkitan yang optimal dengan lebih cepat karena nilai lambda awalnya yang lebih pasti. Dalam tugas akhir ini aplikasi metode *Enhanced Lambda* akan diterapkan pada *software Powergen*. *Powergen* merupakan suatu *software* perhitungan *Economic Dispatch* yang dimiliki Institut Teknologi Sepuluh November berbasis Delphi 7.0. Pengaplikasian metode ini diharapkan dapat membantu proses pembelajaran mata kuliah operasi *optimum* sistem tenaga listrik. Tes validasi dari hasil tugas akhir akan dilakukan dengan mencocokan hasil pembangkitan dari sumber *paper*.

Kata Kunci : *Dynamic Economic Dispatch*, *Enhanced* Lambda *Iterasi, Powergen, Delphi*.

------Halaman ini sengaja dikosongkan----

The Development Of Dynamic Economic Dispatch Software With Considering Transmission Losses Using Enhanced Lambda Iteration

Name ***: Nadia Dwiatmo***
Advisor 1 ***: Prof. Ir. H. Ontoseno Penangsang, M.Sc, Ph.D***
Advisor 2 ***: Dedet Candra Riawan, ST., M.Eng., Ph.D***

ABSTRACT

The economical generation cost is totally support with a good system optimization. Economic dispatch is the short-term determination of the optimal output of a number of electricity generation facilities. Whereas demand of the system load changes in a certain of interval time will be studied in dynamic economic dispatch analysis. The paper proposes an Enhanced Lambda Iteration method for solving the economic dispatch problem in power systems. Enhanced Lambda Iteration is the development methods of lambda iteration which used to gain optimal economic generation value more quickly because the initial value is fixed. The economic dispatch problem is solved by specialized computer software Powergen which should honour the operational and system constraints of the available resources and corresponding transmission capabilities. This software which belong to Institut Teknologi Sepuluh Nopember based on Delphi 7.0. The result are verified with related case in the literature as validation.

***Key Word* : *Dynamic Economic Dispatch*, *Enhanced Lambda Iteration, Powergen, Delphi*.**

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas segala karunia dan bantuan-Nya sehingga penulis dapat menyelesaikan laporan Tugas Akhir dengan judul

"Pengembangan *Software Dynamic Economic Dispatch* dengan Memperhitungkan Rugi-Rugi Transmisi Menggunakan Metode *Enhanced* Lambda Iterasi"

Tugas Akhir ini merupakan salah satu syarat yang harus dipenuhi dalam menyelesaikan Program Studi Strata 1 pada Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember. Dalam proses pengerjaan Tugas Akhir ini penulis telah terbantu oleh beberapa pihak, antara lain:

1. Kedua orang tua, Bapak Trihadi dan Ibu Sudiah, kakak penulis Eky Novianarenti yang selalu mendukung, mendo'akan dan memberikan bantuan berupa material dan non material.
2. Bapak Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D dan Bapak Dedet Candra Riawan, ST., M.Eng., Ph.D selaku dosen pembimbing yang telah memberikan bantuan teknis dan semangat ketika penulis mengalami permasalahan dalam penelitian.
3. Ketua jurusan, Bapak dan Ibu Dosen Pengajar Lintas Jalur Teknik Elektro ITS atas ilmu dan bimbingannya selama proses perkuliahan.
4. Keluarga besar Laboratorium Simulasi Sistem Tenaga Listrik B103, khususnya tim riset *Economic Dispatch* : Santi Triwijaya, Asfar Muhammad, Oktarina Ratri Wijawa dan Silsilatil Maghfiroh.
5. Teman-teman Lintas Jalur Genap 2013, khususnya program studi Teknik *Power* Sistem Tenaga, atas kerja samanya selama masa perkuliahan dan menyelesaikan Tugas Akhir.

Surabaya, Januari 2016

Penulis

-----Halaman ini sengaja dikosongkan----

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

---Halaman Ini Sengaja Dikosongkan---

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Meningkatnya perkembangan pertumbuhan penduduk pada setiap negara menyebabkan jumlah energi yang diperlukan akan semakin meningkat. Peningkatan jumlah kebutuhan energi pada setiap waktu dan wilayah juga akan berbeda-beda sesuai dengan perkembangan pada wilayah tersebut. Perubahan kebutuhan energi tersebut akan berdampak pada perbedaan daya yang harus dibangkitkan pada setiap pembangkit. Dimana hampir 90% pembangkit di indonesia menggunakan pembangkit *thermal*. Salah satu kebutuhan utama untuk suatu pembangkitan unit pembangkit *thermal* adalah biaya untuk bahan bakar. Setiap pembangkit tentunya menginginkan harga pembangkitan termurah namun juga dapat membangkitkan daya listrik yang besar. Dan untuk mendapatkan nilai pembangkitan terefisien maka dilakukan analisis *economic dispatch*. *Economic dispatch* adalah suatu pembagian pembebanan pada unit-unit pembangkit yang ada dalam sistem secara *optimal economi*, pada harga beban sistem tertentu.

Pembebanan kebutuhan sistem tenaga listrik pada setiap daerah tentu akan berbeda. Perbedaan tersebut yang akan menjadikan perbedaan pola konsumsi listrik pada setiap jam. Pembangkit yang efisien adalah pembangkit yang dapat memenuhi kebutuhan beban dengan biaya minimum. Oleh karena itu pada tugas akhir ini akan dibahas *dynamic economic dispatch* menggunakan metode *enhanced lambda* dengan memperhitungkan *losses* pada transmisi. Metode *enhanced lambda* merupakan pengembangan dari metode lambda iterasi. Dimana metode ini merupakan metode klasik yang dapat digunakan juga untuk menyelesaikan analisa permasalahan *economic dispatch* dengan skala besar.

Dalam mengaplikasikan perhitungan analisis ini, penulis merencanakannya dengan menggunakan *software* berbasis delphi. *Software* ini dirancang dengan tampilan *interface* yang mudah dipahami. Diharapkan *software* ini dapat mempermudah *user* dalam menganalisis permasalahan *dynamic economic dispatch*. Sehingga dapat diperoleh distribusi pembangkitan daya paling efisien berdasarkan *economic dispatch* dengan *error minimal*.

1.2. Tujuan Penelitian

Permasalahan yang akan dibahas dalam Tugas Akhir ini adalah :

1. Bagaimana mendapatkan *software* yang dapat *interface* untuk *user*.
2. Bagaimana mendapatkan nilai pembangkitan tentu sesuai dengan *economic dispatch*.
3. Bagaimana mendapatkan nilai pembangkitan dengan menggunakan metode *enhanced lambda* iterasi.

1.3. Batasan Masalah

Agar tugas akhir ini tidak menyimpang dari tujuan dari penelitian maka diambil batasan masalah dan asumsi berikut :

1. Menggunakan metode *enhanced lambda* iterasi untuk optimalisasi.
2. Mempertimbangkan pengaruh rugi-rugi transmisi.
3. Software aplikasi yang digunakan dalam perhitungan yaitu delphi.

1.4. Metode Penelitian

Metode yang digunakan pada tugas akhir ini sebagai berikut :

1. Studi literatur

   Mempelajari literatur mengenai *dynamic economic dispatch*, analisis sistem tenaga, analisa rugi-rugi transmisi, fungsi dan manfaat *economic dispatch* untuk unit pembangkitan, metode-metode analisis *economic dispatch*.

2. Pengumpulan data

   Mengumpulkan data spesifikasi pembangkit dan *losses* dari IET.

3. Pemodelan dan Simulasi

   Memodelkannya dalam bentuk simulasi *software* berbasis delphi untuk beban *dynamic* menggunakan metode analisis *enhanced lambda* iterasi.

3. Analisis data

   Hasil dari simulasi tersebut akan dianalisis seberapa efisien dengan menggunakan metode *enhaned lambda* iterasi dibandingkan dengan metode lainya.

4. Kesimpulan

   Pada proses akhir ini dilakukan pengambilan kesimpulan dari proses tugas akhir.

1.5. Sistematika Penelitian

Sistematika penulisan laporan tugas akhir ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut :

BAB 1 PENDAHULUAN, pada bab ini berisi tentang pendahuluan yang berisi latar belakang, permasalahan, tujuan, metodologi, batasan masalah dan sistematika penulisan.

BAB 2 DASAR TEORI, pada bab ini berisi teori penunjang yang membahas sistem kelistrikan dari pembangkitan sampai konsumen, *economic dispatch*, *dynamic economic dispatch*, *losses*, dan dasar-dasar pemrograman Delphi.

BAB 3 DESAIN DAN SIMULASI , pada bab ini akan berisi tentang proses desain, pemodelan serta simulasi yang dikerjakan agar didapat hasil perhitungan aplikatif *Dynamic Economic Dispatch* (DED). Desain yang dibuat berdasarkan manual dan aplikasi menggunakan *software* Delphi.

BAB 4 HASIL SIMULASI DAN ANALISIS DATA, pada bab ini akan dibahas mengenai hasil simulasi yang dijalankan menggunakan *software* Delphi yang akan dibandingkan dengan metode lain.

BAB 5 PENUTUP, pada bab ini akan ditarik kesimpulan yang dapat diambil dari hasil simulasi yang telah dilakukan analisis. Dan juga saran yang dilampirkan yang diharapkan bisa memberikan perbaikan dan penyempurnaan terkait keberlanjutan tugas akhir ini.

1.6. Manfaat dan Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan beberapa kontribusi untuk perusahaan dan bagi institut pendidikan :

1. Bagi perusahaan listrik :

   Diharapkan dengan tugas akhir ini dapat menjadi pertimbangan dalam pemilihan pola pembangkitan yang dilakukan sehingga didapatkan biaya pembangkitan seminimal mungkin dengan daya pembangkitan yang semaksimal mungkin.
2. Bagi institut pendidikan :

- Mempermudah *user* dalam melakukan perhitungan *Dynamic Economic Dispacth* menggunakan *Enhanced Lambda* Iterasi.
- Menambah penguasaan ilmu dan teknologi di bidang optimalisasi pembangkitan tenaga listrik.
- Sebagai reverensi untuk pengembangan apikasi perhitungan *Dynamic Economic Dispatch*.

---Halaman Ini Sengaja Dikosongkan---

BAB 2
DYNAMIC ECONOMIC DISPATCH PADA SISTEM TENAGA LISTRIK

2.1. Sistem Tenaga Listrik

Energi merupakan kebutuhan yang sangat penting bagi manusia. Setiap individu membutuhkan sumber daya enargi untuk kelangsungan hidupnya. Setiap makhluk hidup, baik individu maupun kelompok berjuang untuk memenuhi kebutuan energinya. Penggunaan energi yang tidak bijak dapat mengakibatkan pengurangan energi secara drastis tanpa diimbangi dengan pelestarian energi tersebut. Semakin langkanya energi maka harga pemenuhanya semakin tinggi, seiring dengan perbandingan kebutuhan energi yang meningkat dan persediaan energi yang menurun.

Sejak ditemukanya energi listrik berabad-abad tahun lalu, menjadikan penemuan ini sebagai energi yang tak habis untuk selalu dikembangkan. Salah satu bentuk energi yang sangat bermanfaat dan tak lekang oleh waktu. Pemanfaataanya yang mudah menjadikan energi listrik salah satu kebutuhan yang tak bisa ditinggalkan. Pada era modern ini banyak pemanfaatan teknolgi berkembang tak luput dari sentuhan energi listrik, baik dimanfaatkan oleh individu maupun perusahaan dan instansi.

Sistem tenaga listrik merupakan upaya penyediaan listrik dari pembangkitan energi listrik sampai pada konsumen. Untuk menunjang hal tersebut maka diperluakan tiga sistem yaitu :

- Sistem pembangkitan.
- Sistem transmisi.
- Sistem distribusi.

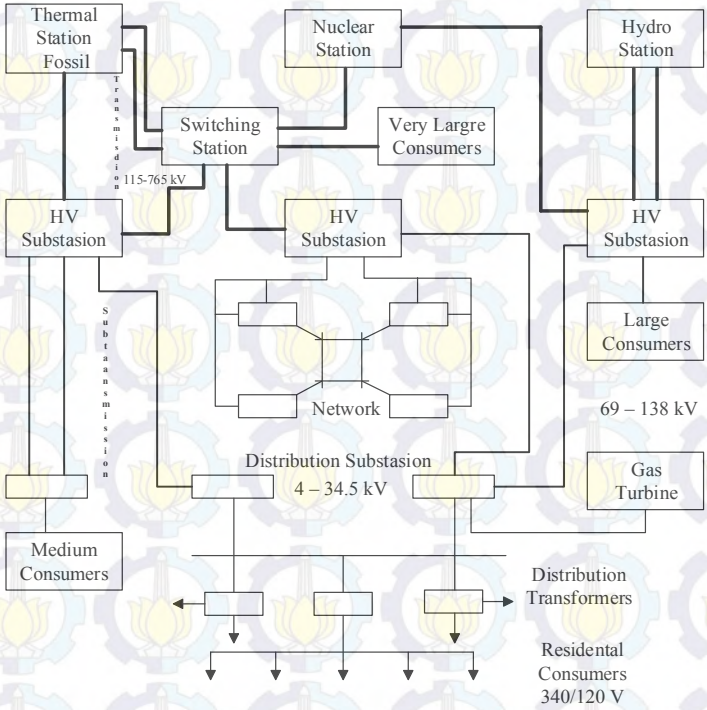
Sistem pembangkitan terdiri dari unit-unit pembangkit tenaga listrik yang terhubung dengan sistem transmisi dan distribusi. Sedangkan sistem transmisi digunakan sebagai penyaluran daya listrik dari sistem pembangkit menuju sistem distribusi dengan menggunakan tegangan tinggi untuk meminimalisir rugi-rugi saluran. Sedangkan fungsi distribusi yaitu sebagai penyalur dari tegangan tinggi transmisi yang diturunkan sesuai dengan tegangan yang dibutuhkan untuk menyuplai daya ke konsumen. Sistem distribusi terdiri dari gardu induk dan beban.

Sistem tenaga listrik modern terdiri dari beberapa pembangkit yang saling terhubung. Sistem tenaga listrik ini disebut dengan sistem interkoneksi. Sistem ini memiliki keuntungan dan kerugian antara lain :

Keuntungan interkoneksi sistem tenanga listrik :

1. Dapat meningkatkan keandalan sistem.
2. Meningkatkan efisiensi pembangkit.
3. Dapat menyalurkan energi listrik ke daerah yang jauh dari sistem pembangkitan.
4. Mempermudah penjadwalan pembangkit.

Berikut adalah contoh skema dasar sistem tenaga listrik modern.



Gambar 2. 1 *Basic Power System* [10]

2.1.1. Pembangkitan

Sebagian besar sistem pembangkitan terdiri dari unit-unit pembangkit tenaga listrik yang terdiri dari beberapa komponen yaitu

turbin dan generator. Tujuan dari pembangkit tenaga listrik yaitu untuk membangkitkan daya listrik yang kemudian ditransmisikan melalui sistem transmisi lalu didistribusikan ke konsumen.

Pembangkit tenanga listrik dibedakan atas beberapa jenis pembangkit, tergantung dari jenis bahan bakar yang digunakan dalam pembangkitanya. Jenis pembangkit tersebut adalah : pembangkit tenaga *thermal* dan pembangkit tenaga panas. Dalam pembangkitanya sumber-sumber dari alam dirubah oleh penggerak mula menjadi energi mekanis yang berupa kecepatan atau putaran, selanjutnya energi mekanis tersebut dirubah menjadi energi listrik oleh generator. Biasanya pada proses pembangkitan listrik oleh generator dihasilkan dengan menggunakan induksi elektromagnetik.

Tiap pembangkit memiliki karakteristik unit pembangkitannya masing-masing. Karakteristik unit pembangkit meliputi karakteristik *input-output* pembangkit dan karakteristik *incremental rate* [1]. Karakteristik tersebut diperoleh dari data-data seperti : desain generator, pabrik pembuat generator, data historis pengoperasian generator, maupun data percobaan. Karakteristik ini yang nantinya digunakan dalam perhitungan biaya pembangkitan dari tiap unit pembangkit sehingga dapat dicapai nilai ekonomis atau optimum. Secara umum karakteristik input output pembangkit *thermal* dalam bentuk *btu/hour*. Biaya pembangkitan adalah perkalian biaya yaitu $, kalori yang terkandung dalam bahan bakar dengan kebutuhan kalori di setiap jam dari generator. Hasil daya yang dibangkitkan berupa *watt* yang direpresentasikan dengan P.

2.1.2. Transmisi

Transmisi sistem tenaga listrik merupakan jaringan tenaga listrik yang berfungsi untuk menyalurkan daya listrik yang dibangkitkan dari sistem pembangkitan menuju sistem distribusi teanga listrik. Pada sistem ini daya listrik akan dinaikan dari daya yang dibangkitkan dengan tujuan untuk mengurangi rugi-rugi jaringan transmisi yang disebabkan oleh panas penghantar akibat adanya arus yang mengalir [2], semakin panjang saluran maka *losses* yang dihasilkan akan semakin besar, *losses* yang diijinkan dalam sistem tenaga listrik ±5% dari tegangan nominal.

Sebagian besar sistem jaringan transmisi yang digunakan di indonesia adalah jaringan transmisi interkoneksi sehingga lebih mudah dalam penyaluran daya pada saat darurat. Tegangan pada sistem transmisi yaitu :

1. Saluran Kabel Tegangan Tinggi (SKTT) : 30kV - 150kV.

2. Saluran Udara Tegangan Tinggi (SUTT) : 30kV - 150kV.
3. Saluran Udara Tegangan Ekstra Tinggi (SUTET) : 200kV - 500kV.

2.1.3. Distribusi

Sistem distribusi berfungsi untuk menyalurkan daya listrik dari sistem transmisi melalui gardu induk distribusi ke peralatan konsumen. Sistem distribusi melayani pelanggan dengan daya besar dan pelanggan *residental*. Dari jenis pelanggan tersebut maka sistem distribusi dibagi menjadi dua yaitu : sistem distribusi primer dan sistem distribusi sekunder. Dimana kapasitas distribusi primer berkisar antara 4kV - 34,5kV. Sedangkan distribusi sekunder akan menyuplai beban *residental* dengan kisaran tegangan antara 220 V / 120 V untuk *single phase* tiga belitan, 220 V / 120 V untuk *three phase* empat belitan, atau 480 V Y/ 277 V *three phase* empat belitan [2].

Sedangkan berdasarkan peletakan penghantar, sistem distribusi dibagi menjadi dua yaitu :

1. *Overhead*.
2. *Underground*.

Saluran *overhead* merupakan kabel atau kawat transmisi listrik yang disalurkan di udara atau diatas tanah. Saluran ini memliki keunggulan yaitu mudah dalam perawatan, perbaikan dan perluasan wilayah. Akan tetapi saluran ini mudah mengalami gangguan akibat lingkungan sekitar. Sedangkan saluran *underground* merupakan kabel yang instalasinya dipasang dibawah tanah. Sistem ini memiliki kelebihan yaitu tidak mudah mendapatkan gangggguan. Namun memiliki biaya pemasangan instalasi dan perawatan yang mahal. Selain itu juga lebih sulit menemukan titik gangguan.

2.1.4. Beban Sistem

Daya yang dihasilkan berupa daya nyata (*watt*). Daya yang disalurkan harus dapat memenuhi permintaan kebutuhan beban. Apabila beban yang dikirimkan tidak sesuai akan menimbulkan masalah berupa kerugian, baik daya itu berlebihan atau kurang dari jumlah permintaan pembebanan. Adapun apabila daya yang dikirimkan berlebih maka akan menimbulkan rugi bagi perusahaan pembangkitan karena pemborosan daya. Sedangkan apabila daya yang dikirimkan kurang dari permintaan maka akan terjadi *over load* dan mengakibarkan pemadaman.

Oleh sebab itu perlu diadakanya peramalan beban untuk dapat memperkirakan berapa daya yang seharusnya diproduksi dan disalurkan.

Hal ini merupakan hal penting bagi perusahaan untuk dapat menentukan teknik operasional yang akan diambil. Metode peramalan beban itu sendiri dibagi menjadi dua yaitu [3] :

1. Metode Kualitatif , metode ini digunakan dimana tidak ada model matematik, biasanya dikarenakan data yang tidak cukup representatif untuk meramalkan masa yang akan datang (*long term forcasting*).
2. Metode Kuantitatif, penggunaan metode ini berdasarkan ketersediaan data mentah disertai serangkaian kaidah matematis untuk meramalkan hasil masa depan.

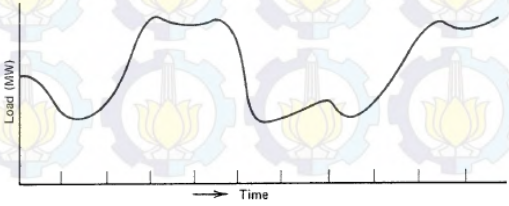
Berdasarkan jangka waktu, perkiraan beban dibagi menjadi tiga yaitu :

1. Perkiraan beban jangka panjang (*long term*), merupakan peramalan beban listrik untuk jangka waktu diatas satu tahun.
2. Perkiraan beban jangka menengah (*medium term*), peramalan beban listrik dengan jangka waktu satu bulan sampai satu tahun.
3. Perkiraan beban jangka pendek (*short term*), peramalan beban listrik untuk jangka waktu beberapa jam sampai satu minggu.

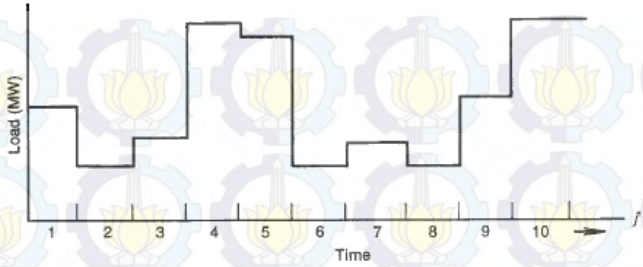
Sedangkan berdasarkan daerah beban dapat dibagi menjadi dua yaitu :

1. Beban industri, industri yang besar memungkinkan untuk mendapatkan suplai daya dari jaringan sub transmisi dan industri kecil mendapatkan pelayanan dari jaringan distribusi primer. Dimana beban industri merupakan beban gabungan yang memiliki fungsi tegangan dan frekuensi
2. Beban komersial dan perumahan, beban jenis ini frekuensi yang relatif tetap dan konsumsi daya yang relatif kecil.

Karakteristik beban akan berbeda-beda sesuai dengan pola konsumsi energi di setiap sektornya. Kurva beban dapat direpresentasikan sebagai gambaran penggunaan daya beban per satuan waktu. Rentang waktu yang digambarkan sesuai dengan kebutuhan beban dan juga jenis konsumen yang digambarkan, baik berupa hari, minggu ataupun tahunan. Kurva beban yang banyak digambarkan dalam satuan daya per hari. Gambar 2.2. menjukan contoh karakteristik pembebanan.



Gambar 2. 2 Kurva Beban [4]



Gambar 2. 3 Kurva Beban Diskrit [4]

2.2. Karakteristik Unit Pembangkit

Karakteristik unit pembangkit yang berbeda-beda menyebabkan setiap unit pembangkit memiliki daya *output* untuk mensuplai beban yang berbeda-beda juga dalam suatu sistem. Dari karakteristik tersebut maka pembangkit digolongkan menjadi tiga unit pembangkit yaitu :

1. pembangkit saat beban dasar (*base load*),
2. pembangkit untuk beban menengah (*load follower*),
3. pembangkit saat beban puncak (*peaker*).

Pembangkit beban dasar (*base load*) memiliki karakteristik pembangkit yang kurang fleksibel, pembangkit ini tidak dapat dihidupkan dan dimatikan dalam waktu yang singkat sehingga pembangkit ini hanya dapat dioprasikan ketika pembangkit telah siap beroperasi. Pembangkit ini biasanya memiliki biaya pembangkitan yang lebih murah dibandingkan dengan yang lain, misalnya PLTU dengan bahan bakar batubara atau pembangkit *hydro* dengan sumber air yang hanya ekonomis jika dioperasikan (tipe *run off river*) [2].

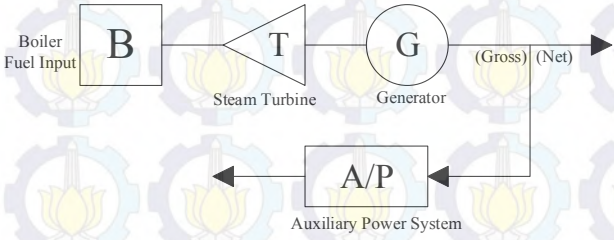
Tipe karakteristik pembangkit untuk beban menengah (*load follower*) lebih fleksibel dari karakteristik beban dasar namun hal itu sebanding dengan biaya pembangkitan yang lebih mahal dari biaya pembangkitan beban dasar. Misalnya PLTGU dan PLTU dengan bahan bakar minyak.

Sedangkan untuk karakteristik pembangkit beban puncak (*peaker*) harus fleksibel, yang mana dalam operasinya memiliki kecepatan perubahan pembebanan ataupun operasi ON/OFF. Pembangkit ini rata-rata berkapasitas dibawah 100MW. Contoh pembangkit dengan

karakteristik yang dapat memikul beban puncak adalah PLTG dengan bahan bakar minyak, PLTA dan PLTD.

2.2.1. Karakteristik *Input-Output* Pembangkit *Thermal*

Fungsi biaya pada dasar perhitungan operasi ekonomis di pembangkit *thermal* ialah berdasarkan karakteristik input-output konsumsi bahan bakar pembangkitnya. Secara umum karakteristik *input-output* pembangkit *thermal* dalam bentuk *btu per hour input* ke unit generator (Mbtu/h). Sedangkan biaya pembangkitan adalah perkalian dari biaya ($) kalori yang terkandung dalam bahan bakar dengan kebutuhan kalori tiap jam dari generator (*btu/h*). Hasil daya yang dibangkitkan berupa *Mega Watt* yang direpresentasikan dengan (P). Tentunya selain biaya bahan bakar yang dikonsumsi, biaya lain juga memerlukan biaya tenaga kerja, biaya pemeliharaan dan perbaikan, biaya transportasi bahan bakar, biaya *improvement*, dan lain-lain. Biaya yang sulit direpresentasikan sebagai fungsi biaya dari daya *output* yang dihasilkan dari generator maka biaya-biaya tersebut diasumsikan sebagai bagian dari *fixed cost* dari biaya operasi [5]. Dalam tugas akhir ini *fixed cost* akan diabaikan.



Gambar 2. 4 Pemodelan *boiler* - turbin - generator [4]

Gambar 2. 4 **Pemodelan *boiler* - turbin - generator [4]**merepresentasikan pembangkit *thermal* sederhana yang terdiri dari *boiler*, turbin uap dan generator. Dimana *input boiler* adalah bahan bakar yang akan menghasilkan uap untuk memutar turbin. Turbin tersebut terkoneksi dengan generator yang akan menghasilkan output berupa daya listrik. Karakteristik dari keseluruhan sistem suatu pembangkit dapat diekspresikan dengan menggabungkan karakteristik *input-output* dari *boiler* dan turbin generator. Karakteristik *input-output* diekspresikan

dalam persamaan yang merupakan pendekatan atau linearisasi dari biaya bahan bakar yang masuk terhadap daya *output* yang diperoleh dari generator yang dihasilkan dari beberapa cara :

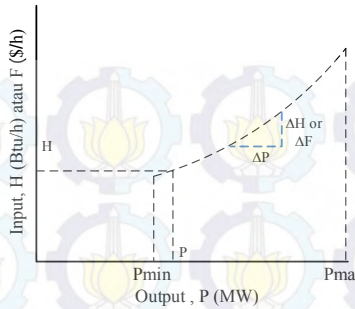
1. Percobaan empiris tentang efisiensi dari pembangkit.
2. Data historis mengenai operasi unit generator.
3. Data desain dari unit generator yang diberikan oleh pabrik pembuat generator.

Persamaan karakteristik *input-output* pembangkit *thermal* tersebut biasanya digambarkan dalam bentuk orde dua, atau bahkan bisa menjadi lebih tidak linier (*non-convex*) apabila memperhatikan pengaruh-pengaruh kecil seperti *valve-point effect*. Sehingga untuk menaganalisis permasalahan mengenai operasi dalam sistem tenaga, terutama masalah dalam operasi ekonomis, dibutuhkan dasar mengenai karakteristik *input-output* dari suatu pembangkit *thermal*. Untuk mendefinisikan karakteristik unit, dibutuhkan penjelasan mengenai *gross input* dan *net output* [5]. Total *input* yang diukur dalam dolar per jam ($/jam) atau kubik gas per jam (gas^3/jam) atau bentuk parameter lain merupakan representasi sebuah *gross input* dari suatu pembangkit. Sedangkan *output* daya listrik yang tersedia untuk penggunaan pada sistem tenaga digambarkan dalam *net output*. Berikut adalah istilah-istilah yang biasanya digunakan dalam mendefinisikan karakteristik dari unit turbin uap :

1. H : Besaran panas sebagai input unit pembangkit (Mbtu/jam).
2. F : Besaran biaya input unit pembangkit (Harga bahan bakar x H) = ($/jam)

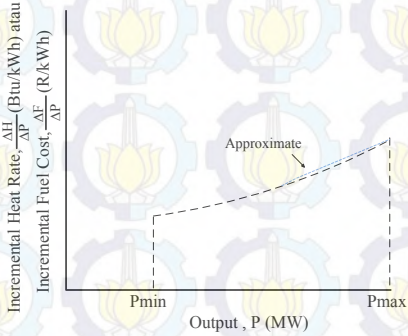
Biaya oprasional ($) tiap jam suatu unit terdiri atas biaya oprasional dan biaya pemeliharaan, biaya pekerja akan dimasukan kedalam biaya operasional. Setiap pembangkit memiliki karakteristik *input-output* dan karakteristik *incremental rate*, dimana *incremental* ini bisa berupa *incremental hear* ataupun *incremental cost*. Gambar 2.6. menunjukan kurva karakteristik *input-output* dari pembangkit *thermal*.

Karakteristik *input-output* dari pembangkit *thermal* merupakan hubungan antara input berupa bahan bakar yang digunakan (Mbtu/h) dengan *output* berupa daya yang dibangkitkan pada tiap pembangkit (MW). Karakteristik kenaikan panas dari unit pembangkit uap digambarkan pada gambar 2.5. dimana kemiringan pada karakteristik *input-output* ($\Delta H/\Delta P$ atau $\Delta F/\Delta P$).



Gambar 2. 5 Karakteristik *Input-Output* Unit Pembangkit [4]

Karakteristik *incremental rate* pembangkit *thermal* merupakan hubungan antara perubahan antara perubahan daya pembangkitan yang dihasilkan dengan konsumsi bahan bakar yang dibutuhkan. Karakteristik ini juga merepresentasikan seberapa besar biaya/panas yang harus ditambahkan saat akan meningkatkan *output* unit pembangkit tersebut. IHR atau *Incremental Heat Rate* (*btu/kWh*) merupakan kurva yang menyatakan gradien kemiringan fungsi *input-output* (ΔH/ΔP) seperti yang ditunjukan pada gambar 2.6.



Gambar 2. 6 Karakteristik *Incremental Rate* [4]

Pemodelan karakteristik *input-output* maupun karakteristik *incremental rate* menggunakan pemodelan *polinominal* dan *piecewise*.

2.2.1.1. Pemodelan Fungsi *Polinominal (Continous)*

Bentuk pemodelan fungsi *polinominal* adalah pendekatan kurva *input-output* dengan fungsi *polinominal*. Kurva *polinominal* yang

biasanya digunakan ialah kurva *polinominal* orde dua. Misalnya pemodelan kurva *polinominal* seperti yang tergambar pada gambar 2.3. merupakan pernyataan dari fungsi persamaan orde dua seperti persamaan 2.1.

$$H\,(P) = aP^2 + bP + c \tag{2.1}$$

Sedangkan fungsi *incremental rate* bisa didapatkan dari turunan fungsi *input-output* pada persamaan 2.2.

$$ihr = \frac{\delta H\,(P)}{\delta P} = 2aP + b \tag{2.2}$$

2.2.1.2. Pemodelan *Piecewise Incremental Heat*

Fungsi *piecewise* adalah fungsi yang didefinisikan oleh sub bab yang digunakan pada interval/segmen yang berbeda. Pemodelan bentuk ini menyajikan serangkaian set data dari kurva *incremental heat* yang kemudian dapat didefinisikan ke dalam bentuk *polinominal* untuk setiap interval/segmen.

Dimana untuk segmen antara titik (X_1 , Y_1) dan (X_2 , Y_2) dapat dibentuk persamaan *polinominal*-nya pada persamaan 2.3.

$$ihr(P) = \propto P + \beta \tag{2.3}$$

Dimana :

$$\propto = \frac{Y_2 - Y_1}{X_2 - X_1} \tag{2.4}$$

$$\beta = \frac{X_2Y_1 - X_1Y_2}{X_2 - X_1} \tag{2.5}$$

Sehingga persamaanya menjadi

$$ihr\,(P) = \frac{Y_2 - Y_1}{X_2 - X_1}P + \frac{X_2Y_1 - X_1Y_2}{X_2 - X_1} \tag{2.6}$$

Dari persamaan 2.6 bisa mendapatkan fungsi *input-output* nya dengan mengintegralkan fungsi ihr.

$$H = \int ihr\,(P)dp = \frac{1}{2} \propto P^2 + \beta P + C \tag{2.7}$$

C adalah bahan bakar minimum saat *output* masih nol *mega watt* atau saat *no load fuel*, atau pada saat fungsi biaya C disebut *no load cost*.

2.3. *Economic Dispatch*

Economic Dispatch (ED) adalah pembagian pembebanan pada pembangkit-pembangkit yang ada secara optimal ekonomi, pada harga beban sistem tertentu. Besar beban pada suatu beban sistem tenaga selalu berubah setiap periode beban tertentu. Karena itu untuk mensuplai beban secara ekonomis perlu dilakukan perhitungan beban *economic dispatch* pada setiap beban. Pengoptimalan permasalahan ED sangat penting untuk melakukan perkiraan jangka panjang dalam sistem tenaga listrik, penentuan porsi biaya, dan pemodelan manageman operasi pembangkit.

Tiga komponen penting dalam dalam pembangkitan energi listrik yaitu biaya pembangunan, biaya kepemilikan, biaya operasional. Biaya operasional merupakan biaya yang berkaitan langsung dengan keuntungan proses produksi karena biaya oprasional berhubungan langsung dengan manageman pembangkitan daya listrik. Setiap pembangkit memiliki nilai pembangkitan yang berbeda karena bergantung dari jenis bahan bakar yang digunakan. Dimanan nilai *fuel cost* sangat mempengaruhi fungsi yang didapat. Persamaan *fuel cost* secara umum dituliskan dalam persamaan 2.8.

$$fuel\ cost = \frac{R}{Mbtu} \tag{2.8}$$

Fuel cost ialah harga per satuan panas dari bahan atau konversi satuan panas ke satuan mata uang. Pengaruh nilai *fuel cost* terhadap fungsi biaya dapat dilihat pada persamaan matematis 2.10. dimana tiap unit ke 'i' akan memiliki persamaan tersebut.

$$Hi\ (Pi) = aiPi^2 + biPi \quad + ci \tag{2.9}$$

$$Fi\ (Pi) = Hi(Pi) \times fuel\ cost\ i \tag{2.10}$$

$$Ftotal = \min \sum_{i=1}^{n} Fi\ (Pi) \tag{2.11}$$

n = Jumlah Generator

Banyaknya unit pembangkit dalam sebuah sistem interkoneksi memberikan kemungkinan pengaturan pembangkitan yang lebih kecil

untuk setiap unit. Batasan yang mempresentasikan keseimbangan daya dalam sistem tercantum dalam *Equality Constrain* pada persamaan 2.12. Sedangkan batasan yang mempresentasikan batasan daya dari pembangkit tercantum dalam persamaan 2.13 yaitu *Inequality Constrain*.

$$\sum_{i=1}^{n} Pi = Pload + Ploss \; n = jumlah \; generator \qquad (2.12)$$

$$Pi \; min \leq Pi \leq Pi \; max \qquad (2.13)$$

$$Ftotal = \min \sum_{i=1}^{n} Fi \; (Pi) \qquad (2.14)$$

n = Jumlah Generator

Interkoneksi antar pembangkit menyebabkan kemungkinan variasi dari pengaturan pembangkit, hal ini yang menyebabkan harga pembangkitan bisa berubah-ubah.

2.4. *Dynamic Economic Dispatch*

Dynamic Economic Dispatch akan menggambarkan kondisi *economic dispatch* dengan mempertimbangkan kondisi *real* beban di lapangan yang akan berubah-ubah sesuai dengan permintaan. Fungsi biaya yang terbentuk akan bergantung dengan permintaan daya beban setiap waktunya. Persamaan 2.15 akan menggambarkan fungsi *heat rate* dari persamaan karakteristik pembangkit. Dan fungsi biaya pembangkitan yang tercantum pada persamaan 2.16.

$$Hi \; (Pi \; (t)) = aiPi^2 + biPi \quad + ci \qquad (2.15)$$

$$Fi \; (Pi(t)) = Hi(Pi(t)) \times fuel \; cost \; i \qquad (2.16)$$

$$Ftotal = \min \sum_{i=1}^{T} \sum_{i=1}^{n} Fi \; (Pi) \qquad (2.17)$$

N = jumlah generator
T = total waktu dalam jam

Dan total biaya Pembangkitan sesuai dengan persamaan 2.15. Demilian juga dengan fungsi *Equality Constrain* 2.18 dan *Inequality Constrain* 2.19 yang ikut mempertimbangkan fungsi t.

$$\sum_{i=1}^{n} Pi(t) = Pload(t) + Ploss\ (t) \quad (2.18)$$

$$Pi\ min\ \leq Pi \leq Pi\ max \quad (2.19)$$

2.5. *Loss Function*

Jaringan transmisi sistem tenaga yang saling interkoneksi akan menimbulkan adanya P *loss* atau rugi daya saluran. Kerugian daya pada transmisi bisa terjadi karena faktor luas penghantar, faktor daya beban dan tegangan *supply*. Rugi-rugi transmisi berbanding lurus dengan besar tahanan konduktor dan berbanding terbalik dengan kuadrat tegangan transmisi, sehingga pengurangan rugi-rugi daya yang diperoleh karena peninggian tegangan transmisi akan lebih efektif jika dibandingkan dengan mengurangi nilai tahanan konduktornya.

Rugi-rugi daya ini harus ditanggung oleh unit pembangkit untuk dapat memenuhi permintaan daya sesuai permintaan beban, sehingga daya yang hilang pada kerugian transmisi akan menjadi beban tambahan pada sistem tenaga.

Rugi-rugi dalam jaringan transmisi sistem tenaga akan menjadi sebuah fungsi pembangkitan. Berdasarkan formula rugi-rugi yang konstan, fungsi disapatkan dalam persaman quadratik yang diselesaikan dengan mengetahui *loss coefficient* atau *B-coefficient* atau *B-loss matrix*. Metode untuk mendapatkan rugi-rugi dan *incremental loss* dalam perhitungan *economic dispatch* sudah diperkenalkan sejak tahun 1950 [7]. Perasamaan *B-coefficient* biasanya terdiri dari matrik [B], matrik B0, dan matrik B00. Persamaan P *loss* didapat dengan fungsi 2.21.

$$P\ loss = [P1 \quad \dots \quad Pn] \begin{bmatrix} B11 & \cdots & B1j \\ \vdots & \ddots & \vdots \\ Bi1 & \cdots & Bij \end{bmatrix} \begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + [Bo1 \quad \dots \quad Boi] \begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + Boo \quad (2.20)$$

Persamaan diatas dapat dituliskan seperti persamaan 2.18

$$P\ loss = P^T[B]P + Bo^T P + Boo \quad (2.21)$$

Dimana :

P : *vector* semua generator (MW)

$[B]$: *matrix* persegi dari dimensi yang sama dengan P

$B0$: vektor dengan panjang yang sama dengan P

$B00$: koefisien konstan

Salah satu metode untuk mendapatkan nilai *B-coefficient* yaitu dengan menggunakan metode Newton-Raphson. Metode ini diciptakan oleh Kron dan diadopsi oleh Kirchmayer[10].

Total daya nyata pada suatu sistem tenaga listrik adalah

$$S_i = P_i + jQ_i = V_i I_i^* \quad (2.22)$$

Dan total rugi daya pada suatu sistem ialah

$$P_L + jQ_L = \sum_{i=1}^{n} V_i I_i^* = V_{bus}^T I_i^* \quad (2.23)$$

Dimana P_L dan Q_L merupakan rugi daya real dan reaktive dari sistem.

$$I_{bus} = Y_{bus} \times V_{bus} \quad (2.24)$$

$$V_{bus} = Y_{bus}^{-1} \times I_{bus} \quad (2.25)$$

$$V_{bus} = Z_{bus} \times I_{bus} \quad (2.26)$$

Persamaan 2.26 disubstitusikan ke persamaan 2.23.

$$P_L + jQ_L = [Z_{bus} I_{bus}]^T I_{bus}^* \quad (2.23)$$

Dimana $V_{bus}^T = Z_{bus}$ maka

$$P_L + jQ_L = V_{bus}^T Z_{bus} I_{bus}^* \quad (2.24)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} I_i \, Z_{ij} \, I_{bus}{}^{*} \qquad (2.25)$$

Rugi daya pada persamaan 2.25 kemudian dipecah menjadi real dan imajiner.

$$P_L = \sum_{i=1}^{n} \sum_{j=1}^{n} I_i \, R_{ij} \, I_j{}^{*} \qquad (2.26)$$

Dalam bentuk matrix menjadi

$$P_L = I_{bus}^{T} R_{bus} \, I_{bus}{}^{*} \qquad (2.27)$$

R_{ij} merupakan elemen real dari matrix impedansi bus. Mencari total arus beban pada sistem tersebut

$$I_{L1} + I_{L2} + \cdots I_{Lnd} = I_D \qquad (2.28)$$

Dimana :

nd = Jumlah Arus Beban

I_D = Total Arus Beban

Asumsikan arus beban individual memiliki pembagian yang sama, sehingga didapat nilai arus:

$$I_{Lk} = \ell_k \, I_D \qquad k = 1,2, \ldots, nd \qquad (2.29)$$

Atau

$$\ell_k = \frac{I_{Lk}}{I_D} \qquad (2.30)$$

Jika diasumsikan bus 1 itu adalah Slack bus, maka dari persamaan 2 didapat:

$$V_1 = Z_{11}I_1 + Z_{12}I_1 + \cdots Z_{1n}I_n \quad (2.31)$$

Jika n_g adalah jumlah bus pembangkit dan jumlah bus beban, maka persamaan 2.31 menjadi

$$V_1 = \sum_{i=1}^{ng} Z_{1i}I_{gi} + \sum_{k=1}^{nd} Z_{1k}I_{Lk} \quad (2.32)$$

Substitusi I_{Lk} ke persamaan 2.29 dan 2.32

$$V_1 = \sum_{i=1}^{ng} Z_{1i}I_{gi} + I_D \sum_{k=1}^{nd} \ell_k Z_{1k} \quad (2.33)$$

$$= \sum_{i=1}^{ng} Z_{1i}I_{gi} + I_D T \quad (2.34)$$

Dimana :

$$T = \sum_{k=1}^{nd} \ell_k Z_{1k} \quad (2.35)$$

Jika I_0 didefinisikan sebagai arus yang keluar dari bus 1 dan dianggap semua arus beban sama dengan nol maka nilai tegangan menjadi

$$V_1 = -Z_{11}I_0 \quad (2.36)$$

Substitusi persamaan 2.36 ke persamaan 2.34

$$V_D = -\frac{1}{T}\sum_{i=1}^{ng} Z_{1i}I_{gi} - \frac{1}{T} Z_{1i}I_0 \quad (2.37)$$

Substitusi ke persamaan 2.29 sehingga arus beban menjadi

$$I_{Lk} = -\frac{\ell_k}{T}\sum_{i=1}^{ng} Z_{1i} I_{gi} \; - \frac{\ell_k}{T}\; Z_{1i} I_0 \tag{2.37}$$

Jika $\rho = -\frac{\ell_k}{T}$ maka (2.38)

$$I_{Lk} = \rho k \sum_{i=1}^{ng} Z_{1i} I_{gi} \; + \; \rho k \, Z_{1i} I_0 \tag{2.39}$$

Apabila persamaan 2.39 dirubah menjadi persamaan matrik sehingga menjadi

$$\begin{bmatrix} I_{g1} \\ I_{g2} \\ \\ I_{gn_g} \\ I_{L1} \\ I_{L2} \\ \\ I_{Ln_d} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ & & & \\ 0 & 0 & 1 & 0 \\ \rho_1 Z_{11} & \rho_1 Z_{12} & \rho_1 Z_{1n_g} & \rho_1 Z_{11} \\ \rho_2 Z_{11} & \rho_2 Z_{12} & \rho_2 Z_{1n_g} & \rho_2 Z_{11} \\ & & & \\ \rho_k Z_{11} & \rho_k Z_{12} & \rho_k Z_{1n_g} & \rho_k Z_{11} \end{bmatrix} \begin{bmatrix} I_{g1} \\ I_{g2} \\ \\ I_{gn_g} \\ \\ \\ \\ I_0 \end{bmatrix} \tag{2.40}$$

Persamaan matrik diatas dapat disingkat menjadi C sehingga

$$I_{bus} = \; C I_{new} \tag{2.41}$$

Substitusi persamaan 2.41 ke persamaan 2.27

$$P_L = [C I_{new}]^T R_{bus} \, C^* {I_{new}}^* \tag{2.42}$$

$$= I_{new}^T C^T R_{bus} \, C^* {I_{new}}^* \tag{2.43}$$

Jika S_{gi} adalah daya *complex* bus 1 maka arus pembangkitnya menjadi

$$I_{gi} = \frac{{S_{gi}}^*}{{V_i}^*} = \frac{P_{gi} - j\, Q_{gi}}{{V_i}^*} = \frac{1 - j\,\frac{Q_{gi}}{P_{gi}}}{{V_i}^*}\; P_{gi} \tag{2.44}$$

$$I_{new} = \psi P_{gi} \tag{2.45}$$

Substitusi persamaan 2.45 ke persamaan 2.43

$$P_L = \left[\psi P_{gi}\right]^T C^T R_{bus}\, C^* \psi^* {P_{gi}}^* \tag{2.46}$$

$$= P_{G1}^T \psi^T C^T R_{bus}\, C^* \psi^* {P_{gi}}^* \tag{2.47}$$

$$= P_{G1}^T \Re[H] {P_{gi}}^* \tag{2.48}$$

Persamaan diatas jika dijadikan kedalam bentuk matrik menjadi

$$\Re[H] = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1ng} & B_{01}/2 \\ B_{21} & B_{22} & \cdots & B_{1ng} & B_{01}/2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{ng1} & B_{ng2} & \cdots & B_{ngng} & B_{0ng}/2 \\ B_{01}/2 & B_{01}/2 & \cdots & B_{0ng}/2 & B_{00} \end{bmatrix} \tag{2.49}$$

$$P_L = [P_{g1} \quad P_{g2} \quad \cdots \quad P_{gng} \quad 1] \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1ng} & B_{01}/2 \\ B_{21} & B_{22} & \cdots & B_{1ng} & B_{01}/2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{ng1} & B_{ng2} & \cdots & B_{ngng} & B_{0ng}/2 \\ B_{01}/2 & B_{01}/2 & \cdots & B_{0ng}/2 & B_{00} \end{bmatrix} \begin{bmatrix} P_{g1} \\ P_{g2} \\ \vdots \\ P_{gng} \\ 1 \end{bmatrix} \tag{2.50}$$

Jika dijabarkan menjadi bentuk [B], B0 dan B00 menjadi

$$P_L = [P_{g1} \quad P_{g2} \quad \cdots \quad P_{gng}] \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1ng} \\ B_{21} & B_{22} & \cdots & B_{1ng} \\ \vdots & \vdots & \ddots & \vdots \\ B_{ng1} & B_{ng2} & \cdots & B_{ngng} \end{bmatrix} \begin{bmatrix} P_{g1} \\ P_{g2} \\ \vdots \\ P_{gng} \end{bmatrix} + [P_{g1} \quad P_{g2} \quad \cdots \quad P_{gng}] \begin{bmatrix} B_{01} \\ B_{02} \\ \vdots \\ B_{0ng} \end{bmatrix} + B_{00} \tag{2.51}$$

Pada fungsi *dynamic economic dispatch* nilai rugi-rugi generator dapat dicari dengan menggunakan persamaan 2.22

$$P\ loss = P^T[B]P + Bo^T P + Boo \tag{2.52}$$

$$P\ loss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo \qquad (2.53)$$

Dimana,
P : *vector* semua generator (MW)
$[B]$: *matrix* persegi dari dimensi yang sama dengan P
$B0$: vector dengan panjang yang sama dengan P
$B00$: koefisien konstan

Penalty factor pada *economic dispatch* tanpa *losses* sama dengan satu. Sedangkan pada perhitungan *economic dispatch* dengan mempertimbangkan rugi-rugi daya maka akan menyebabkan nilai *penalty factor* pada setiap unit berbeda-beda sesuai dengan persamaan 2.24. Nilai *penalty factor* akan mempengaruhi nilai dari lambda.

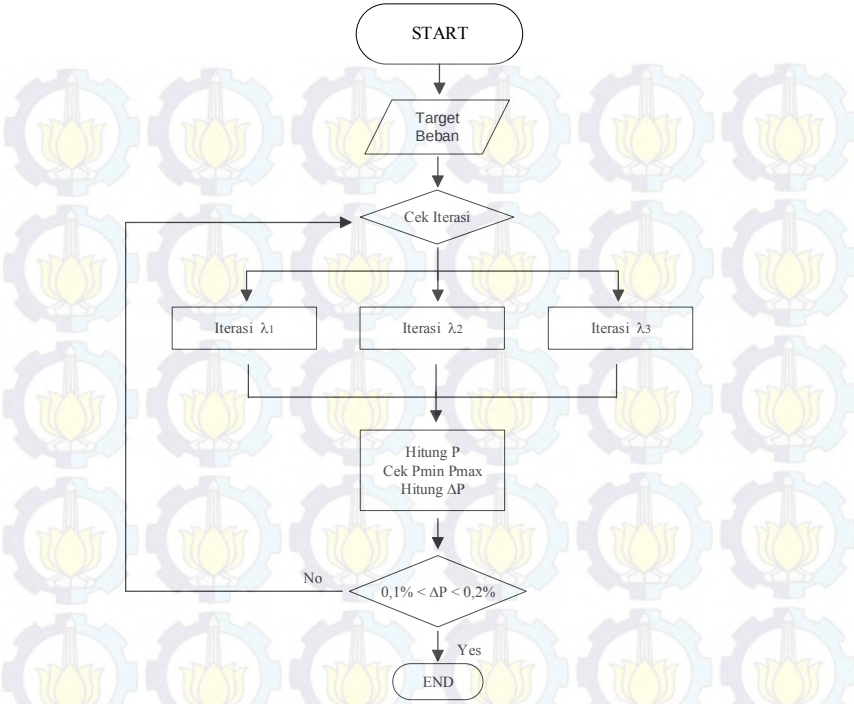
$$Pf = \frac{1}{1 - \frac{\partial P\ loss}{\partial Pi}} \qquad (2.54)$$

2.6. *Enhanced Lambda*

Pada tugas akhir ini akan digunakan metode *enhanced lambda* untuk menyelesaikan persamaan *Dynamic Economic Dispatch*. *Enhanced lambda* merupakan metode pengembangan dari iterasi lambda. Metode ini nilai lambda awal tidak ditentukan dengan *try and error* tetapi dicari dengan menggunakan persamaan 2.25. diharapkan dengan penetapan nilai lambda yang tepat maka akan didapatkan nilai yang maksimal dengan *gencost* yang ekonomis dan jumlah iterasi yang singkat.

$$\lambda^1 = \frac{P_D + \sum_{i=1}^{N} \frac{b_i}{2ci}}{\sum_{i=1}^{N} \frac{1}{2a_i}} ; i = 1,2,3 \ldots n \qquad (2.55)$$

Dimana,
P_D: Target generator (MW)



Gambar 2. 7 *Flow Chart* Pengerjaan Lambda Iterasi

Persamaan diatas didapat dengan persamaan berikut

$$P_L = \sum_{i=1}^{N} \frac{\lambda - b_i}{2a_i} \; ; i = 1,2,3 \ldots n \tag{2.56}$$

$$P_L + \sum_{i=1}^{N} \frac{b_i}{2a_i} = \lambda \left(\frac{1}{2a_1} + \frac{1}{2a_1} + \cdots + \frac{1}{2a_n} \right) \tag{2.57}$$

Sehingga didapat nilai lambda sesuai dengan persamaan 2.55 diatas.

Dengan nilai F seperti persamaan 2.16. Dari persamaan 2.55 maka dapat dicari nilai pembangkitan tiap generator dengan menggunakan persamaan 2.58 metode enhanced lambda menggunakan *galat tolerance* sebesar ±(0.1-0,2%). *Error* tersebut dicari dengan nilai ΔP pada persamaan 2.59.

$$P_i{}^1 = \frac{\lambda^1 + b_i}{2C_i}\,; i = 1,2,3 \ldots n \tag{2.58}$$

$$\Delta P^1 = P_D - \sum_{i=1}^{N} P_i{}^1 \tag{2.59}$$

Jika nilai ΔP^1 telah memunuhi syarat toleransi maka iterasi dinyatakan selesai. Namun jika nilai ΔP^1 tidak memnuhi batasan nilai *error* maka cari nilai λ^2 dengan persamaan 2.60.

$$\lambda^2 = \pm 10\,\% \times \lambda^1 \tag{2.60}$$

Kenaikan ±10% bergantung pada nilai ΔP^1, jika ΔP^1 bernilai negatif maka +10% dan sebaliknya. Demikian selanjutnya seperti langkah sebelumnya setelah didapat nilai lambda dilanjutkan mencari nilai $P_i{}^2$ dan ΔP^2. Dan jika masih belum memenuhi syarat toleransi maka dicari lambda berikutnya dengan menggunakan persamaan 2.9 berikut.

$$\lambda^3 = \frac{\lambda^2 \times \Delta P^1 - \lambda^1 \times \Delta P^2}{\Delta P^1 - \Delta P^2} \tag{2.61}$$

$$P_i{}^3 = \frac{\lambda^3 + b_i}{2C_i}\,; i = 1,2,3 \ldots n \tag{2.62}$$

$$\Delta P^3 = P_D - \sum_{i=1}^{N} P_i{}^3 \tag{2.63}$$

Hitung nilai $P_i{}^3$ dengan menggunakan persamaan 2.62 dan ΔP^3 pada persamaan 3.63. Kalkulasi dan analisa data hasil perhitungan apakah telah memenuhi syarat *error* atau belum. Jika belum maka lakukan berulang dengan mengiterasi lambda dengan persamaan 2.61 secara terus menerus hingga didapatkan nilai ΔP sesuai dengan target yang ditentukan sebelumnya.

Lambda yang ketiga dicari dengan menggunakan prinsip gradien

$$m = \frac{y_2 - y_1}{X_2 - X_1} \tag{2.64}$$

Dengan mengasumsikan nilai y = 0 maka didapat nilai lambda yang ketiga pada persamaan 2.65.

$$\lambda^{baru} = \frac{\lambda^{2+1} \times \Delta P^{1+1} - \lambda^{1+1} \times \Delta P^{2+1}}{\Delta P^{1+1} - \Delta P^{2+1}} \tag{2.65}$$

$$P_i^{baru} = \frac{\lambda^{baru} + b_i}{2C_i}; i = 1,2,3 \dots n \tag{2.66}$$

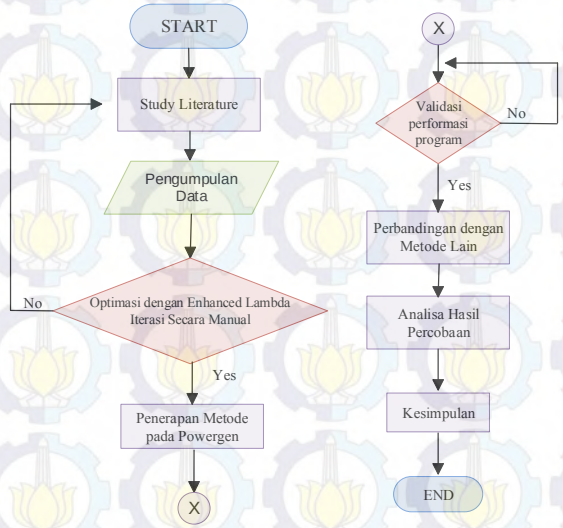
$$\Delta P^{baru} = P_D - \sum_{i=1}^{N} P_i^{baru} \tag{2.67}$$

Persamaan diatas sama dengan mengganti nilai λ sebelumnya dengan λ baru, demikian juga dengan nilai ΔP. Iterasi ini terus berulang sampai ditemukan nilai ΔP = ±(0.1-0,2%).

BAB 3
PERANCANGAN *DYNAMIC ECONOMIC DISPATCH* DENGAN ENHANCED LAMBDA PADA SISTEM STANDARD IEEE 6 BUS

3.1. Perancangan Algoritma DED

Dalam perancangan tugas akhir ini digambarkan pada *flow chart* berikut.



Gambar 3. 1 *Flow Chart* Pengerjaan Tugas Akhir

Flow chart diatas menjelaskan alur pengerjaan tugas akhir yang akan disusun. Tugas akhir ini dimulai dengan studi literatur tentang *economic dispatch* dan rugi-rugi transmisi. Setelah didapat metode dan data karakteristik yang tepat maka dilakukan perhitungan manual dengan menggunakan metode tersebut. Metode yang akan digunakan dalam tugas akhir ini adalah metode *enhanced lambda* iterasi. Untuk dapat melakukan perhitungan manual. Perhitungan manual ini juga dapat dijadikan sebagai acuan dalam kebenaran progran yang akan dibuat nanti.

Setelah program yang dibuat diuji dengan perbandingan perhitungan manual dan sumber-sumber data lainya sebagai referensi dengan metode selain *enhanced lambda* iterasi dianggap OK maka lanjut ke proses selanjutnya. Proses selanjutnya adalah dilakukan analisis data dan penarikan kesimpulan dari percobaan tugas akhir ini.

3.2. *Powergen*

Powergen merupakan perangkat lunak yang dibuat oleh Teknik Elektro Institut Teknologi Sepuluh November yang digunakan untuk melakukan perhitungan berkaitan dengan sistem tenaga listrik. *Software* ini juga digunakan sebagai *tools* untuk alat bantu dalam proses akademik oprasi optimum sistem tenaga listrik.

Dalam *software* ini terdapat beberpa menu yang dapat digunakan untuk membantu menyelesaikan pemasalahan sistem tenaga listrik. Antara lain :

1. *Power Flow* : Untuk melakukan perhitungan aliran daya.
2. DUBLP : Digunakan untuk melakukan perhitungan OOSTL dengan menggunakan *linear programming*.
3. EDC : Menu ini dilakukan untuk melakukan perhitungan *economic dispatch*.
4. HYDRO : Digunakan untuk menyelesaikan permasalahan penjadwalan pembangkit tenaga air / *hydro*.
5. Unitcom : *Tools* untuk menyelesaikan permasalahan *Unit commitment*.
6. DED : Menu untuk menyelesaikan permasalahan *Dynamic Economic Dispatch*.
7. CEED : Digunakan untuk menyelesaikan permasalahan *economic dispatch* dan *emmision dispatch*.
8. UC+ED : Unit Commitment yang di Dispatch
9. EXIT : Menu keluar dari program

Gambar tampilan menu utama pada *software powergen* akan digambarkan pada gambar 3.2. Dalam tugas akhir yang dirancang akan membahas menu *Dynamic Economic Dispatch* dengan menggunakan penyelesaian *Enhanced Lambda* Iterasi.

Gambar 3. 2 Menu Tampilan Utama *Software Powergen*

3.2.1. Sintak *Input* DED

Sintak program adalah perintah yang digunakan untuk memanggil program dengan argumen *input* yang dimasukkan. Oleh karena itu diperlukan beberapa sintak yang dapat berkaitan dengan operasi menu *enhanced lambda* iterasi. Berikut beberapa sintak yang dapat digunakan antara lain :

1. datadump : Untuk memasukan semua data awal perhitungan di setiap periode yang akan ditampilan pada lembar *output*.
2. datainput : Menerima masukan data *file* yang telah tersimpan.
3. dataoutput : Menulis masukan pada data *file* agar dapat tersimpan.
4. ihr_ftn : Inisiasi turunan pertama persamaan Hi (Pi(t)) sebagai pengali dengan fungsi *fuelcost*, untuk mendapatkan inisiasi $\frac{\partial Fi}{\partial Pi}$
5. inverse_ihr_ftn : Mencari nilai pembangkitan setiap unit ketika diperoleh nilai lambda.
6. lambda_search_dispatch : Untuk menjalankan *economic dispatch* dengan menmakai metode lambda *search*.
7 loss_matrix_ftn : Untuk memperhitungkan nilai *losses* pada sistem.

8.	enhanced_lambda	:	Digunakan untuk menghitung *economic dispatch* dengan menggunakan metode *enhanced lambda* iterasi.
9.	output_routine	:	Prosedur yang terdiri dari urutan untuk memperoleh hasil akhir pada tiap periode.
10.	output_final	:	Untuk memproses dan menampilkan hasil akhir dari *output* yang diinginkan.

3.2.2. Argumen *Input Output* untuk DED

Arguman merupakan masukan awal dalam bahasa Delphi yang digunakan dalam perhitungan DED. Argumen yang akan digunakan dalam operasi *enhanced lambda* antara lain dijelaskan sebagai berikut:

1.	coeff[i]	:	Sebagai masukan awal dari nilai koefisien A, B, C dalam persamaan karakteristik pembangkitan Hi(Pi(t)).
2.	fuelcost[i]	:	Masukan awal dari *fuel cost* yang akan digunakan pada persamaan Fi(Pi(t)).
3.	fuelprice[i]	:	Masukan berupa harga bahan bakar per satuan jumlah volume.
4.	Ratting[i]	:	*Rating thermal* bahan bakar yang digunakan oleh unit pembangkit.
5.	loadjam[i]	:	Masukan nilai pembebanan per satuan periode.
6.	data_period[i]	:	Sebagai data *array* dari total biaya pembangkitan masing-masing pada setiap periode.
7.	total_period[i]	:	Merupakan total data *array* dari total biata pembangkitan per satuan periode.
8.	finalqty	:	Total keseluruhan penggunaan bahan bakar.
9.	numofload	:	Merupakan masukan berapa banyak dari jumlah pembebanan.

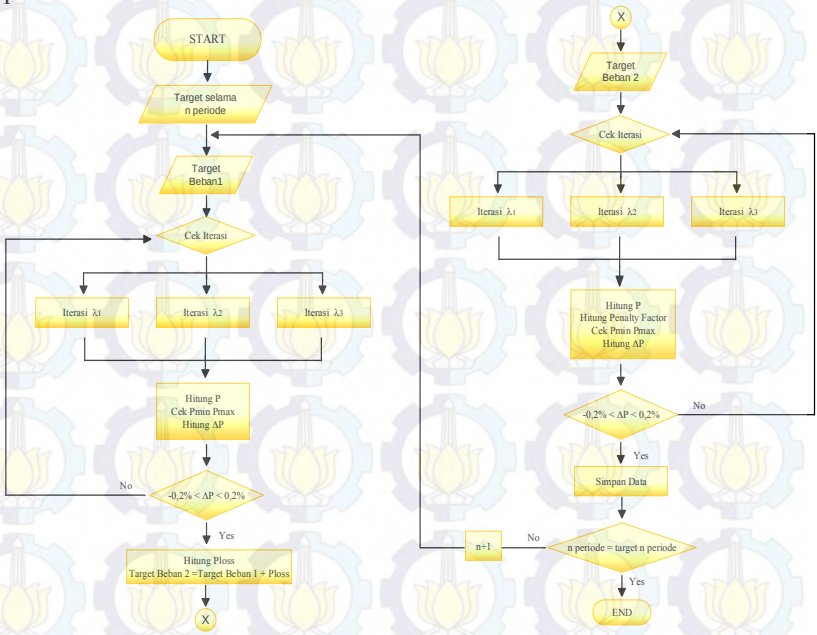
3.3. Perancangan DED dengan Metode *Enhanced Lambda* Iterasi

Pada perancangan dibagi menjadi dua langkah, yaitu pertama perencanaan secara matematis dan perencanaan yang diimplementasikan

pada program delphi. Perencanaan secara matematis dilakukan sebagai validasi dari program yang dibuat.

3.3.1. Perhitungan Matematis *Enhanced Lambda* Iterasi

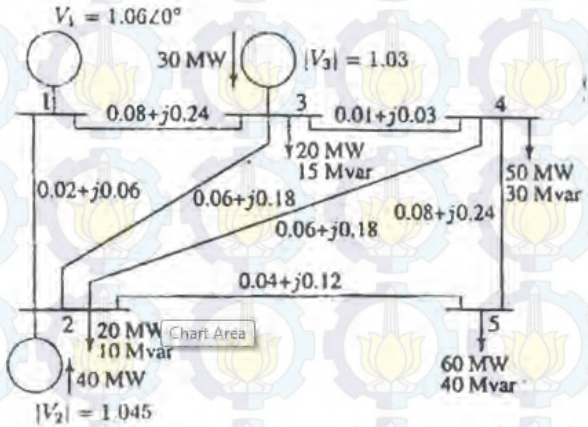
Perhitungan secara matematis dengan menggunakan metode *Enhanced Lambda* Iterasi dilakukan dengan tiga pokok langkah dalam mencari lambda. Hal tersebut yang menjadi pembeda dengan metode lambda iterasi karena dapat menentukan nilai lambda dengan lebih cepat. Dengan penentuan nilai awal lambda yang tepat maka akan mempercepat proses iterasi. Sedangkan pada lambda iterasi nilai awal dari lambda ditentukan dengan menebak angka lambda. Proses menebak nilai lambda ini yang bisa saja menyebabkan proses iterasi yang lebih banyak. Hal tersebut karena tebakan nilai awal lambda bisa saja jauh dari nilai lambda yang optimal. *Flow chart* proses pengerjaan iterasi lambda telah dijelaskan pada BAB 2. Berikut *flow chart* pengerjaan perhitungan losses pada Gambar 3. 3 ***Flow Chart*** **Metode** ***Enhanced Lambda*** **Iterasi**.



Gambar 3. 3 *Flow Chart* Metode *Enhanced Lambda* Iterasi

3.3.1.1. Kasus Pertama dengan Tiga Pembangkit

Pada kasus pertama akan digunakan tiga pembangkit. Kasus ini mempertimbangkan nilai *losses* transmisi yang dihitung dengan menggunakan formula *B-coefficient*. *Base* MVA kasus pertama ialah 100 MVA. Berikut data yang dibutuhkan dalam perhitungan kasus pertama:



Gambar 3. 4 Diagram Satu Garis Plan 3 Pembangkit 6 Bus

Tabel 3. 1 Data Beban

No Bus	Load	
	MW	Mvar
1	0	0
2	20	10
3	20	15
4	50	30
5	60	40

Tabel 3. 2 Data Generator

No Bus	*Voltage Magnitude*	Generator (MW)	Mvar Limits	
			Min	Max
1	1,06	0	0	0
2	1,045	40	10	50
3	1,03	30	10	40

Tabel 3. 3 Data Line dan Transformer

No Bus	No Bus	R(pu)	X(pu)	1/2B(pu)
1	2	0,02	0,06	0,03
1	3	0,08	0,24	0,025
2	3	0,06	0,18	0,02
2	4	0,06	0,18	0,02
2	5	0,04	0,12	0,015
3	4	0,01	0,03	0,01
4	5	0,08	0,24	0,025

Data Karakteristik untuk Kasus Pertama

$$F_1 = 200 + 7P_1 + 0{,}008{P_1}^2$$
$$F_2 = 180 + 6{,}3P_1 + 0{,}009{P_1}^2$$
$$F_3 = 140 + 6{,}8P_1 + 0{,}007{P_1}^2$$

Tabel 3. 4 Generator Limit

Pmin	Pmax
10	85
10	80
10	70

Tabel 3. 5 Data *B-Coefficient* untuk Kasus Pertama

		0,0218	0,0093	0,0028
B	:	0,0093	0,0228	0,0017
		0,0028	0,0031	0,0015
B0	:	0,0003	0,0031	0,0015
B00	:	0,00030523		

Pada kasus pertama ini dihitung dengan beban pembangkitan 150MW sebagai referensi pembanding dengan data yang sebelumnya telah dihitung.

- Menghitung lambda tanpa *losses*
  Menggunakan persamaan 2.25 didapat nilai lambda :

$$\lambda^1 = \frac{P_D + \sum_{i=1}^{N} \frac{b_i}{2ci}}{\sum_{i=1}^{N} \frac{1}{2a_i}}$$

$$\lambda^1 = \frac{150 + \left(\frac{7}{2 \times 0{,}008} + \frac{6{,}3}{2 \times 0{,}009} + \frac{6{,}8}{2 \times 0{,}007}\right)}{\left(\frac{1}{2 \times 200} + \frac{1}{2 \times 180} + \frac{1}{2 \times 140}\right)} = 7{,}51099476$$

- Menghitung total pembangkitan tanpa *losses*
  Menggunakan persamaan 2.26 didapat nilai total pembangkitan :

$$P_i^{\,1} = \frac{\lambda^1 - b_i}{2C_i}; i = 1,2,3 \ldots n \qquad (2.26)$$

$$P_1 = \frac{7{,}510995 - 7}{2 \times 0{,}008} = 31{,}93717$$

$$P_2 = \frac{7{,}510995 - 6{,}3}{2 \times 0{,}009} = 67{,}27749$$

$$P_3 = \frac{7{,}510995 - 6{,}8}{2 \times 0{,}007} = 50{,}78534$$

$$\sum_{i=1}^{N} P = P_1 + P_2 + P_3 = 150$$

- Menghitung *Equality Constrain* pembangkitan tanpa *losses*
  Menggunakan persamaan 2.27 didapat nilai ΔP^1 :

$$\Delta P^1 = P_D - \sum_{i=1}^{N} P_i^{\,1} \qquad (2.27)$$

$$\Delta P^1 = 150 - 150 = 0$$

- Menghitung losses pembangkitan
  Setelah didapat $\Delta P^1 = 0$ dengan tanpa *losses* kemudian hitung nilai P*loss* dengan menggunakan nilai pembangkit tersebut. Persamaan P*loss* dapat dilihat pada persamaan 2.23 yang dijelaskan pada bab sebelumnya:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo$$

$$Ploss = [P1 \quad P2 \quad P3]\begin{bmatrix} B11 & B12 & B13 \\ B21 & B22 & B23 \\ B31 & B32 & B33 \end{bmatrix}\begin{bmatrix} P1 \\ P2 \\ P3 \end{bmatrix} + [Bo1 \quad Bo2 \quad Bo3]\begin{bmatrix} P1 \\ P2 \\ P3 \end{bmatrix} + Boo$$

$$Ploss = [31{,}93717 \quad 67{,}27749 \quad 50{,}78534]\begin{bmatrix} 0{,}0218 & 0{,}0093 & 0{,}0028 \\ 0{,}0093 & 0{,}0228 & 0{,}0017 \\ 0{,}0028 & 0{,}0031 & 0{,}0015 \end{bmatrix}\begin{bmatrix} 31{,}93717 \\ 67{,}27749 \\ 50{,}78534 \end{bmatrix} + [0{,}0003 \quad 0{,}0031 \quad 0{,}0015]\begin{bmatrix} 31{,}93717 \\ 67{,}27749 \\ 50{,}78534 \end{bmatrix} + 0{,}00030523 = 0{,}026475 \text{ PU}$$

$$Ploss = 0{,}02648 \times 100 = 2{,}6475 \text{ MW}$$

- Target pembangkitan baru
  Target pembangkitan yang lama ditambahkan dengan nilai *losses* transmisi :

  PD = PD lama + *Losses* = 150 + 2,6475 = 152,6475 MW

- Menghitung *penalty factor* pembangkitan #Iterasi=1
  Menggunakan persamaan perhitungan 2.24

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf = \frac{1}{1 - 2\sum_j B_{ij}P_j - B_{ij}} = \frac{1}{1 - A}$$

$$A = \left(2 \times \begin{bmatrix} 0{,}0218 & 0{,}0093 & 0{,}0028 \\ 0{,}0093 & 0{,}0228 & 0{,}0017 \\ 0{,}0028 & 0{,}0031 & 0{,}0015 \end{bmatrix} [31{,}93717 \quad 67{,}27749 \quad 50{,}78534] \right) + [0{,}0003 \quad 0{,}0031 \quad 0{,}0015]$$

$$Pf_1 = \frac{1}{1 - 0{,}03018219} = 1{,}0267$$

$$Pf_2 = \frac{1}{1 - 0{,}04764555} = 1{,}0378$$

$$Pf_3 = \frac{1}{1 - 0{,}026757068} = 1{,}0213$$

- Menghitung lambda dengan *losses* #Iterasi=1
  Menggunakan persamaan 2.25 didapat nilai lambda :

$$\lambda^1 = \frac{P_D + \sum_{i=1}^{N} \frac{b_i}{2ci}}{\sum_{i=1}^{N} \frac{1}{2a_i}}$$

$$\lambda^1 = \frac{152{,}648 + \left(\frac{7}{2 \times 0{,}008} + \frac{6{,}3}{2 \times 0{,}009} + \frac{6{,}8}{2 \times 0{,}007} \right)}{\left(\frac{1}{2 \times 200} + \frac{1}{2 \times 180} + \frac{1}{2 \times 140} \right)} = 7{,}52497$$

- Menghitung total pembangkitan dengan *losses* #Iterasi=1
  Menggunakan persamaan 2.26 didapat nilai total pembangkitan :

$$P_i{}^1 = \frac{\lambda^1 - b_i}{2C_i} ; i = 1,2,3 \ldots n \qquad (2.26)$$

$$P_1 = \frac{7{,}52497 - 7 \times 1{,}0267}{2 \times 0{,}008 \times 1{,}0267} = 20{,}5817$$

$$P_2 = \frac{7{,}52497 - 6{,}3 \times 1{,}0378}{2 \times 0{,}009 \times 1{,}0378} = 52{,}8286$$

$$P_3 = \frac{7{,}52497 - 6{,}8}{2 \times 0{,}007} = 40{,}5757$$

$$\sum_{i=1}^{N} P = P_1 + P_2 + P_3 = 113{,}9861$$

$$\Delta P^1 = P_D - \sum_{i=1}^{N} P_i{}^1 \qquad (2.27)$$

- Menghitung *Equality Constrain* pembangkitan tanpa *losses* #Iterasi=1 Menggunakan persamaan 2.27 didapat nilai ΔP^1 :

$$\Delta P^1 = P_D - \sum_{i=1}^{N} P_i{}^1 \qquad (2.27)$$

$$\Delta P^1 = 152 - 113{,}9861 = 38{,}6614$$

Pada iterasi pertama didapatkan nilai dari pembangkitan sebesar 113,9861 MW. Dengan demikian maka ada selisih pembangkitan dengan penjadwalan pembangkitan sebesar 38,6614 MW dan perlu dilakukan iterasi kembali hingga didapat nilai pembangkitan yang memiliki *equality constrain* antara 0,1% - 0,2%.

Pengerjaan iterasi berikutnya sama dengan langkah pengerjaan pada iterasi satu. Untuk plan kasus satu dengan target generator sebesar 150 MW didapat dengan melakukan iterasi sebanyak 35 kali iterasi. Berukut hasil akhir perhitungan untuk kasus pertama.

- Menghitung *losses* pembangkitan #Iterasi = 35
  Setelah didapat $\Delta P^1 = 0$ dengan tanpa *losses* kemudian hitung nilai P*loss* dengan menggunakan nilai pembangkit tersebut. Persamaan P*loss* dapat dilihat pada persamaan 2.23 yang dijelaskan pada bab sebelumnya:

$$Ploss = \sum_{i}^{n} \sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo$$

$Ploss$

$$= [0{,}334654 \quad 0{,}640933 \quad 0{,}550962] \begin{bmatrix} 0{,}0218 & 0{,}0093 & 0{,}0028 \\ 0{,}0093 & 0{,}0228 & 0{,}0017 \\ 0{,}0028 & 0{,}0031 & 0{,}0015 \end{bmatrix} \begin{bmatrix} 0{,}334654 \\ 0{,}640933 \\ 0{,}550962 \end{bmatrix}$$

$$+ [0{,}0003 \quad 0{,}0031 \quad 0{,}0015] \begin{bmatrix} 0{,}640933 \\ 0{,}640933 \\ 0{,}550962 \end{bmatrix} + 0{,}00030523 = 0{,}026683 \text{ PU}$$

$$\text{P}loss = 0{,}02648 \times 100 = 2{,}6683 \text{ MW}$$

- Target pembangkitan baru #Iterasi = 35
  Target pembangkitan yang lama ditambahkan dengan nilai *losses* transmisi :
  PD = PD lama + *Losses* = 150 + 2,6678 = 152,6687 MW

- Menghitung *penalty factor* pembangkitan #Iterasi=33
  Menggunakan persamaan perhitungan 2.24

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf = \frac{1}{1 - 2\sum_j B_{ij} P_j - B_{ij}} = \frac{1}{1 - A}$$

$$A = \left(2 \times \begin{bmatrix} 0{,}0218 & 0{,}0093 & 0{,}0028 \\ 0{,}0093 & 0{,}0228 & 0{,}0017 \\ 0{,}0028 & 0{,}0031 & 0{,}0015 \end{bmatrix} [0{,}334654 \quad 0{,}640933 \quad 0{,}550962] \right) + [0{,}0003 \quad 0{,}0031 \quad 0{,}0015]$$

$$Pf_1 = 1{,}0308$$

$$Pf_2 = 1{,}0421$$

$$Pf_3 = 1{,}0259$$

- Menghitung lambda dengan losses #Iterasi=33
  Menggunakan persamaan 2.25 didapat nilai lambda :

$$\lambda^{35} = \frac{\lambda^{34} \times \Delta P^{33} - \lambda^{33} \times \Delta P^{34}}{\Delta P^{33} - \Delta P^{34}}$$

$$\lambda^{35} = \frac{7{,}7678 \times (-0{,}0021) - 7{,}7678 \times 0{,}0011}{-0{,}0021 - 0{,}0011} = 7{,}7678$$

- Menghitung total pembangkitan dengan *losses* #Iterasi=33
  Menggunakan persamaan 2.26 didapat nilai total pembangkitan :

$$P_i{}^1 = \frac{\lambda^1 - b_i}{2C_i}; i = 1,2,3 \ldots n \qquad (2.26)$$

$$P_1 = 33{,}4201$$

$$P_2 = 64{,}0906$$

$$P_3 = 55{,}0972$$

$$\sum_{i=1}^{N} P = P_1 + P_2 + P_3 = 152{,}6686$$

$$\Delta P^1 = P_D - \sum_{i=1}^{N} P_i{}^1 \qquad (2.27)$$

- Menghitung *Equality Constrain* pembangkitan tanpa *losses* #Iterasi=33
  Menggunakan persamaan 2.27 didapat nilai ΔP^1 :

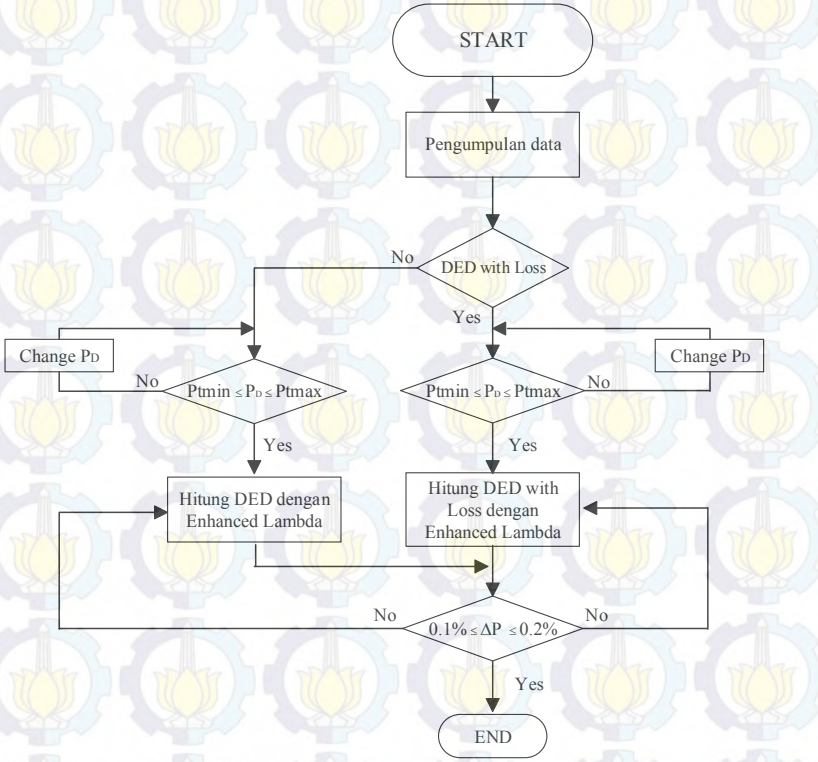
$$\Delta P^1 = P_D - \sum_{i=1}^{N} P_i{}^1 \qquad (2.27)$$

$$\Delta P^1 = 152{,}6687 - 152{,}6686 = 0{,}0001$$

Nilai selisih target generator sudah seuai dengan target yang ditentukan yaitu 0,1% dengan nilai akhir dari pembangkitan sebesar 152,6686 MW. Dimana PD = 150 MW dan P*loss* = 2,6686 MW.

3.3.2. Penerapan *Enhanced Lambda* pada *Software* Delphi 7.0
Dalam pengaplikasian metode *enhanced lambda* pada *software* delphi 7.0 dilakukan dengan *flow chart* sebagai pada gambar

Gambar 3. 5 **Skema Perencanaan Algoritma Pemrograman DED**Dalam tugas akhir ini direncanakan perhitungan *dynamic economic dispatch* dengan mempertimbangkan rugi-rugi transmisi. Perhitungan DED akan dihitung dengan pertimbangan menyerupai beban pembebanan yang mendekati nilai pada keadaan sebenarnya. Beban yang digunakan dibuat dengan pembebanan 1x24jam dengan perubahan beban yang bervariasi.



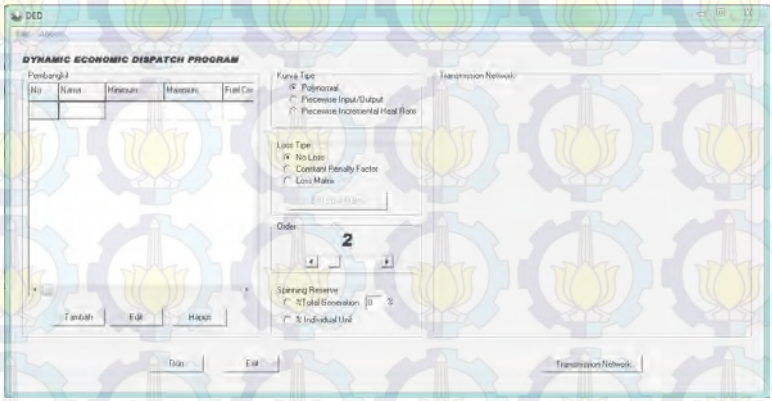
Gambar 3. 5 Skema Perencanaan Algoritma Pemrograman DED

Pengaplikasian penyelesaian masalah DED dimulai dengan pengumpulan data karakteristik pembangkit, data *losses* dan beban yang

ditargetkan. Program aplikasi yang dibuat dapat menganalisis perhitungan ketika data diketahui dengan atau tanpa *losses* transmisi. Dari analisis tersebut akan dilakukan perhitungan DED menggunakan *Enhanced Lambda* Iterasi dengan target pembebanan sesuai dengan *limit* dari kemampuan generator. Hasil akhir akan dicapai jika *error* dari ΔP sesuai. Penyelesaian permasalahan akan diprogram pada aplikasi Delphi 0.7.

3.3.2.1. Desain Menu Utama DED *Powergen*

Langkah pertama dilakukan adalah mengumpulkan data yang diperlukan dalam perhitungan analisis DED. Data yang akan digunakan antara lain berupa karakteristik pembangkit, *fuel cost*, *losses* transmisi,dan lain sebagainya. Data tersebut dimasukan dalam *form* utama DED.



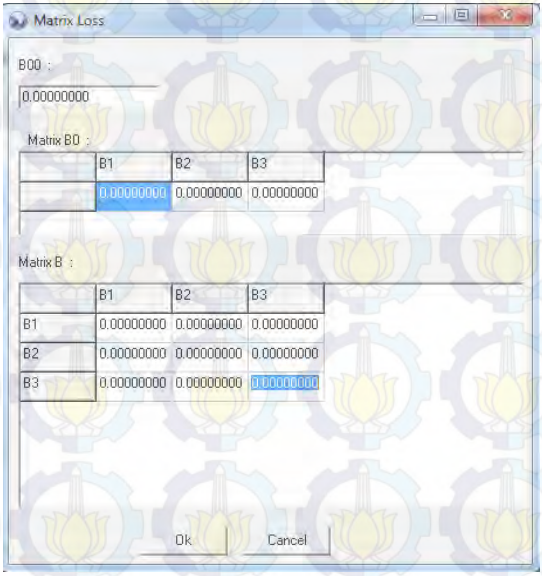
Gambar 3. 6 Tampilan Awal Menu DED

Desain tampilan pada *software powergen* yang akan digunakan memiliki beberapa fitur seperti yang telah dijelaskan pada gambar 3.2 diatas. Tampilan gambar 3.5 merupakan tampilan dari fitur menu DED. Dalam *form menu* 'Pembangkit' tersebut terdapat menu 'Tambah' , 'Edit', 'Hapus' dan beberapa kolom yang menunjukkan nama unit pembangkit, *min* dan *max* pembangkit dan harga bahan bakar. Form ini didesan sebagai menu untuk memasukan nilai-nilai dari karakteristik pembangkitan. Dimana nilai dari karakteristik pembangkitan merupakan data yang diperlukan dalam perhitungan *economic dispatch*.

Pada menu kurva *type* terdapat tiga jenis kurva yaitu *polinomial, piecewise input/output, piecewise incremental heat rate*. Di

tugas akhir ini hanya akan membahas kasus dengan jenis kurva *polynomial*.

Loss type menu digunakan untuk menentukan dalam perhitungan analisis *dynamic economic dispatch* ini akan dilakukan dengan atau tidak mempertimbangkan rugi transmisi. Pada tugas akhir ini analisis *economic dispatch* dapat diperhitungkan baik dengan atau tanpa *losses* transmisi. Rugi transmisi yang akan diperhitungkan dengan cara perhitungan *b-coefficient*. Data *b-coefficient* dapat di*edit* pada menu *edit loss* matrik seperti gambar 3.5.

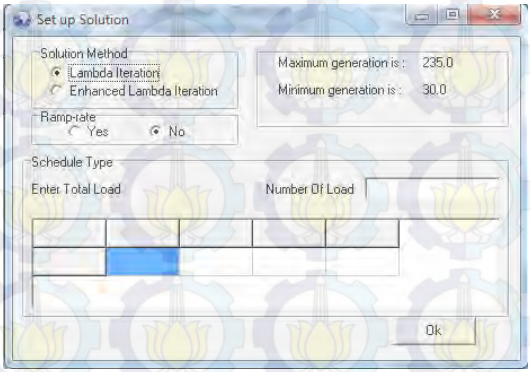


Gambar 3. 7 Tampilan Menu *Matrix losses*

Button Transmission Network digunakan untuk menampilkan gambar *single line diagram* sistem transmisi yang digunakan. Merupakan optional jika memang tidak perlu diperlihatkan.

Pada menu dialog *Set-up Solution* dirancang untuk penjadwalan dengan target beban yang diinginkan pada menu '*Schedule Type*', metode yang digunakan pada menu '*Solution Method*' batasan *ramp rate* pada menu '*Ramp-rate*'. Pada dialog ini juga ditampilkan

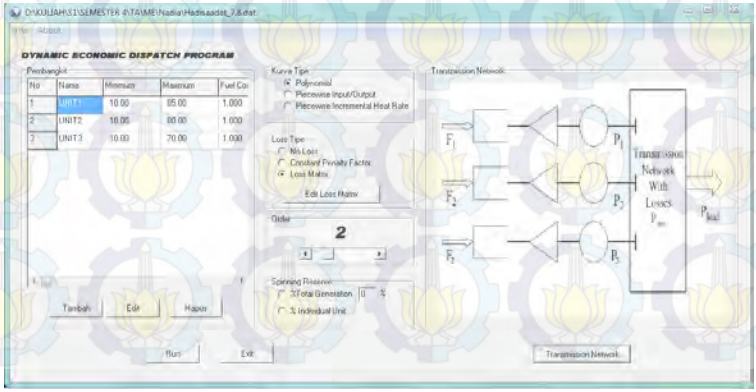
batasan *minimum* dan *maximum* yang dapat dibangkitkan oleh semua unit pembangkit.



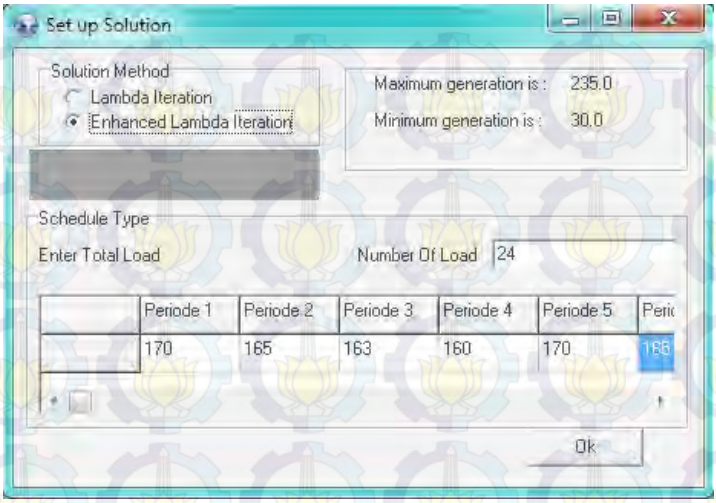
Gambar 3. 8 Tampilan Menu *Set-up Solution* pada *Powergen*

3.3.2.2. Menjalankan Program DED pada *Powegen*

Setelah semua data yang diperlukan untuk perhitungan DED dimasukan ke *software powergen* klik '*Run*' untuk menjalankan program. Tahap selanjutnya ialah menentukan pembebanan yang akan dijadwalkan selama 24 jam. Pada menu dialog *Set-up Solution* menggunakan metode *enhanced lambda* iterasi tanpa mempertimbangkan batasan *ramp rate*. Gambar 3.8 menggambarkan contoh *set-up* penjadwalan yang diinginkan.

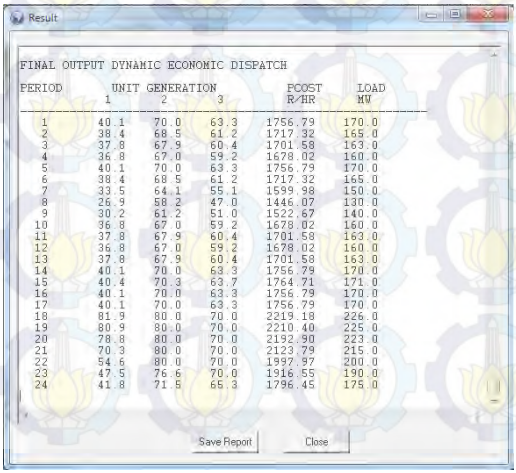


Gambar 3. 9 Contoh Penjalanan Program pada *Powergen*



Gambar 3. 10 Contoh *Set-up* Penjadwalan Beban 24 Jam

Tekan tombol 'OK' untuk konfirmasi proses analisis perhitungan DED. Hasil *running program* dapat dilihat pada gambar 3.9 berikut.



Gambar 3. 11 Contoh Hasil '*Result*' Simulasi Program *Powergen*

BAB 4
IMPLEMENTASI *SOFTWARE DYNAMIC ECONOMIC DISPATCH* DENGAN ENHANCED LAMBDA

4.1. Implementasi *Software*

Bab ini akan membahas hasil dari simulasi *economic dispatch* dengan menggunakan metode *enhanced lambda* iterasi yang diaplikasikan menggunakan s*oftware powergen* berbasis delphi. Pembahasan uji kasus pada tugas akhir ini akan dilakukan dengan meggunakan tiga kasus. Dimana kasus ini terdiri dari tiga pembangkit, enam pembangkit dan sepuluh pembangkit.

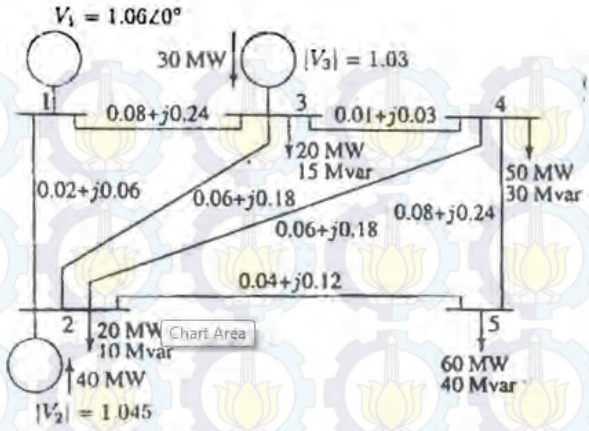
Ketiga kasus tersebut merupakan kasus *dynamic economic dispatch* dengan memperhitungkan rugi-rugi transmisi. Simulasi ini dilakukan dengan periode waktu selama 24 jam yang mengacu pada kurva beban jawa bali. Tujuan dari simulai ini agar dikondisikan sedekat mungkin dengan keadaan *real* di kenyataan dan diharapkan didapat nilai yang lebih baik dari sebelumnya.

4.2. Pengujian Kasus DED

Uji validasi perlu dilakukan untuk mengetahui ketepatan dari pengaplikasian metode *enhanced lambda* pada *software*. Pengujian akan dilakukan dalam dua kasus, yang pertama kasus dengan menggunakan 3 unit pembangkit dan 6 bus. Dan kedua kasus dengan menggunakan 6 unit pembangkit dengan 26 bus.

4.2.1. Kasus DED Pertama dengan Tiga Pembangkit

Pada kasus pertama akan digunakan tiga pembangkit. Kasus ini mempertimbangkan nilai *losses* transmisi yang dihitung dengan menggunakan formula *B-coefficient*. *Base* MVA kasus pertama ialah 100 MVA. Berikut data yang dibutuhkan dalam perhitungan kasus pertama :



Gambar 4. 1 Diagram Satu Garis Kasus Pertama

Tabel 4. 1 Data Beban

No Bus	Load	
	MW	Mvar
1	0	0
2	20	10
3	20	15
4	50	30
5	60	40

Tabel 4. 2 Data Generator

No Bus	*Voltage Magnitude*	Generator (MW)	Mvar Limits	
			Min	Max
1	1,06	0	0	0
2	1,045	40	10	50
3	1,03	30	10	40

Tabel 4. 3 Data transformer

No Bus	No Bus	R(pu)	X(pu)	1/2B(pu)
1	2	0,02	0,06	0,03
1	3	0,08	0,24	0,025
2	3	0,06	0,18	0,02

Tabel 4. 3 Data transformer

No Bus	No Bus	R(pu)	X(pu)	1/2B(pu)
2	4	0,06	0,18	0,02
2	5	0,04	0,12	0,015
3	4	0,01	0,03	0,01
4	5	0,08	0,24	0,025

Tabel 4. 4 Data Karakteristik untuk Kasus Pertama

Unit	a	b	c	P_{min}	P_{max}
1	200	7	0,008	10	85
2	180	6.3	0,009	10	80
3	140	6.8	0,007	10	70

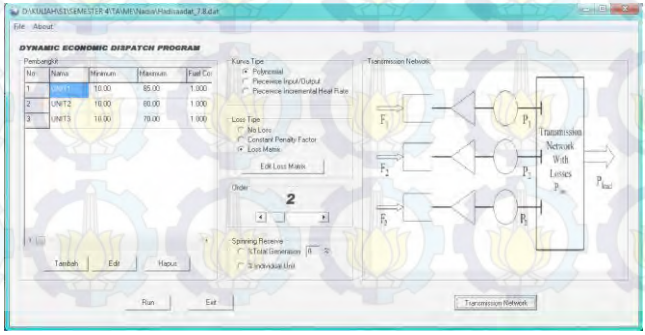
Data *B-Coefficient* untuk Kasus Pertama

$$B : \begin{matrix} 0.0218 & 0.0093 & 0.0028 \\ 0.0093 & 0.0228 & 0.0017 \\ 0.0028 & 0.0031 & 0.0015 \end{matrix}$$

$$B0 : \begin{matrix} 0.0003 & 0.0031 & 0.0015 \end{matrix}$$

$$B00 : 0.00030523$$

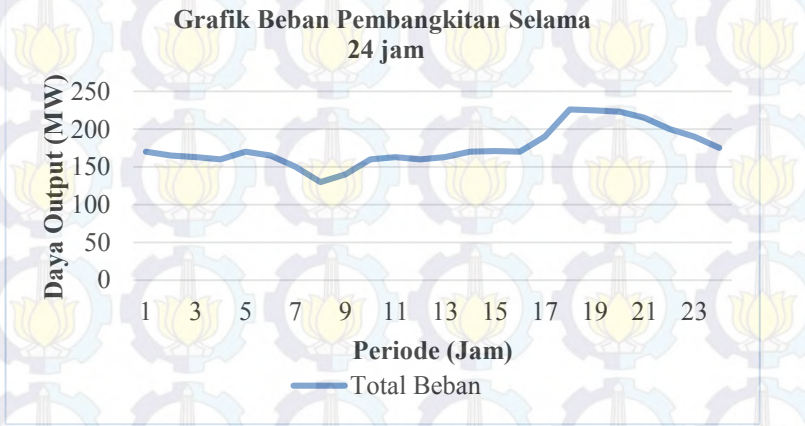
Pada kasus pertama ini dihitung dengan beban pembangkitan selama 24 jam dengan interval waktu pada setiap periodenya selama 1 jam. Beban pembangkitan yang digunakan tercantum pada tabel 4.3. Proses penjalanan program akan ditampilkan pada gambar 4.2.



Gambar 4. 2 Proses Pemasukan Data pada *Software Powergen*

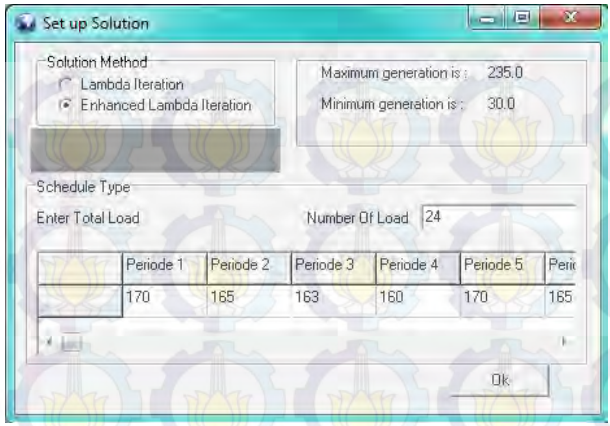
Tabel 4. 5 Beban Pembangkitan selama 24 Jam Kasus Pertama

Beban	Jam	Beban	Jam
170	1	163	13
165	2	170	14
163	3	171	15
160	4	170	16
170	5	190	17
165	6	226	18
150	7	225	19
130	8	223	20
140	9	215	21
160	10	200	22
163	11	190	23
160	12	175	24



Gambar 4. 3 Target Total Pembangkitan dalam 24 Jam

Dilakukan running program menggunkan metode *enhanced lambda* iterasi maka didapat hasil pembagian daya pada generator pada tabel 4.2. Proses penjalanan program dapat dilihat pada gambar 4.4.



Gambar 4. 4 Pemilihan *Set Up Target* dan Metode yang Digunakan pada *Powergen*

Tabel 4. 6 Pembagian Daya Pembangkit Kasus 1 Selama 24 jam

Periode	Unit Generator			Generator *Cost* (Rp/HR)	*Load* (MW)
	1	2	3		
1	40,09	70,00	63,26	1756.7902	170
2	38,44	68,53	61,21	1717.3159	165
3	37,77	67,93	60,40	1701.5774	163
4	36,78	67,05	59,17	1678.0241	160
5	40,09	70,00	63,26	1756.7902	170
6	38,44	68,53	61,21	1717.3159	165
7	33,47	64,10	55,10	1599.9835	150
8	26,87	58,21	46,99	1446.0666	130
9	30,17	61,15	51,04	1522.6662	140
10	36,78	67,05	59,17	1678.0241	160
11	37,77	67,93	60,40	1701.5774	163
12	36,78	67,05	59,17	1678.0241	160
13	37,77	67,93	60,40	1701.5774	163
14	40,09	70,00	63,26	1756.7902	170
15	40,42	70,30	63,66	1764.7079	171
16	40,09	70,00	63,26	1756.7902	170

Tabel 4. 6 Pembagian Daya Pembangkit Kasus 1 Selama 24 jam

Periode	Unit Generator			Generator *Cost* (Rp/HR)	*Load* (MW)
	1	2	3		
17	47,52	76,62	70,00	1916.5541	190
18	81,94	80,00	70,00	2219.1814	226
19	80,88	80,00	70,00	2210.4012	225
20	78,77	80,00	70,00	2192.9045	223
21	70,33	80,00	70,00	2123.7941	215
22	54,60	80,00	70,00	1997.9729	200
23	47,52	76,62	70,00	1916.5541	190
24	41,75	71,48	65,30	1796.4499	175

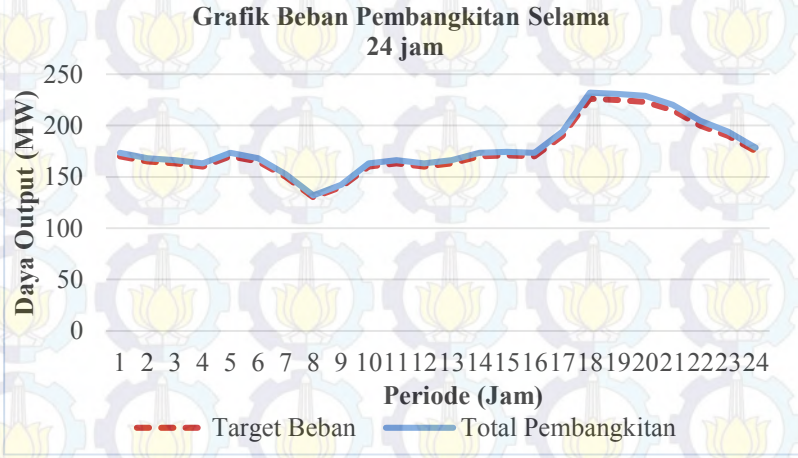
Pada kasus satu ini juga terdapat rugi-rugi transmisi yang menyebakan nilai pembangkitan harus lebih besar dari target. Karena $\sum_{i=1}^{N} Pi = \sum_{i=1}^{N} P_D + Pl$. Tabel 4.5 menunjukan penambahan *losses* pada setiap pembangkit. Sedangkan *Result* pada *software Powergen* tercantum pada gambar 4.5 dibawah.

Tabel 4. 7 Total Pembangkitan Kasus 1

Periode	P_{losses}	PD	P_{total}	No Iterasi
1	3,353	170	173,353	53
2	3,174	165	168,174	50
3	3,104	163	166,104	43
4	3,000	160	163,000	35
5	3,353	170	173,353	53
6	3,174	165	168,174	50
7	2,669	150	152,669	35
8	2,069	130	132,069	39
9	2,358	140	142,358	56
10	3,000	160	163,000	35
11	3,104	163	166,104	43
12	3,000	160	163,000	35
13	3.104	163	166,104	43
14	3.353	170	173,354	53
15	3,390	171	174,389	47
16	3,353	170	173,353	53

Tabel 4. 7 Total Pembangkitan Kasus 1

Periode	P_{losses}	PD	P_{total}	No Iterasi
17	4,141	190	194,141	47
18	5,939	226	231,941	31
19	5,881	225	230,881	39
20	5,767	223	228,767	38
21	5,332	215	220,332	34
22	4,603	200	204,603	23
23	4,141	190	194,141	47
24	3,538	175	178,538	36



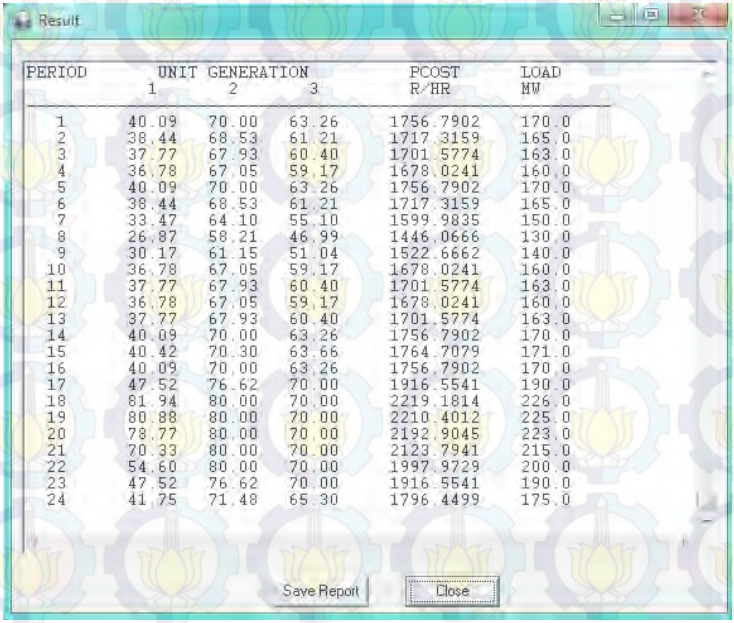
Gambar 4. 5 Perbandingan Daya yang Dibangkitkan dengan Total Beban Target

Dari grafik diatas dapat terlihat nilai pembangkitan dari generator hampir mendekati nilai targer beban total. Hal ini terjadi karena adanya penambahan nilai *losses*. Sehingga nilai *output*nya berubah. Jumlah dari iterasi menggunakan *enhanced lambda* iterasi lebih singkat jika dibandingkan dengan metode iterasi lambda. Berikut gambar hasil simulasi yang dilakukan pada *software Powergen* dapat dilihat pada gambar 4.6.

Untuk melakukan validasi hasil simulasi maka dilakukan perbandingan hasil simulasi *powergen* dengan contoh *plan* yang ada pada pada referensi [10]. **Tabel 4. 6** menunjukan hasil perbandingan antara *Lamba search*, *Newton-Raphson*, *Enhanced Lambda* Iterasi.

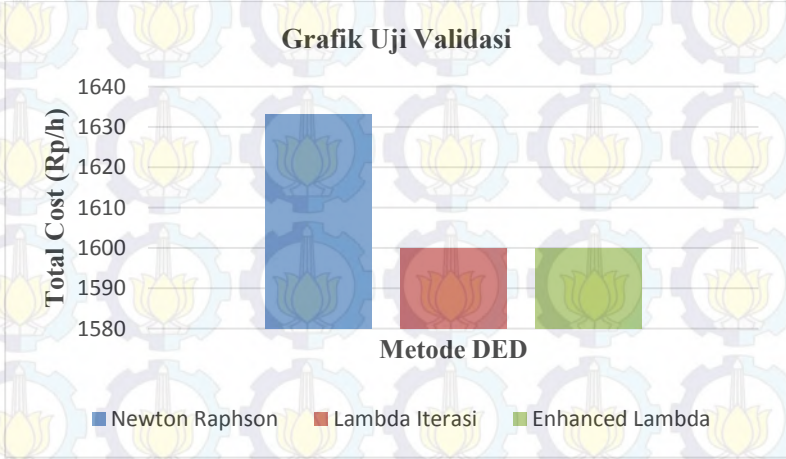
Tabel 4. 8 Perbandingan Hasil Simulasi Kasus 1

Metode / *Variable*	Metode		
	Newton-Raphson	***Lambda***	***Enhanced Lambda***
P1 (MW)	83,051	33,4698	33,4701
P2 (MW)	40,000	64,0977	64,0974
P3 (MW)	30,000	55,1012	55,1011
P_{total} (MW)	153,0510	152,6688	152,6687
P_{loss} (MW)	3,05348	2,6687	2,6687
***Total Cost* (Rp/h)**	**1633.237**	**1599.842**	**1599.9835**



Gambar 4. 6 Tampilan Hasil Akhir pada *Software Powergen*

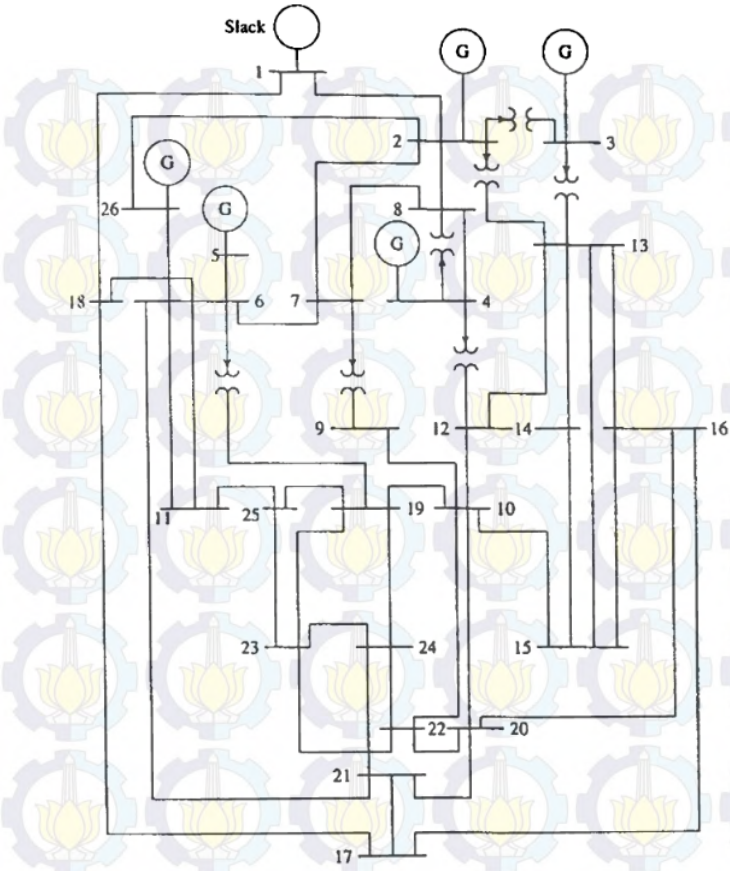
Uji validasi dilakukan dengan menggunakan daya pembangkitan sebesar 150MW. Dengan *base* MVA sebesar 100MVA. Dari hasil simulasi diatas menunjukan bahwa nilai pembangkitan pada setiap generator hampir sama atau mendekati sama. Biaya pembangkitan *enhanced lambda* iterasi lebih murah jika dibandingkan dengan kedua metode lainya. Selain itu *enhanced lambda* iterasi mencapai nilai konvergen lebih cepat dengan 35 iterasi. Sedangkan lambda iterasi mencapai nilai lambda yang konvergen selama ±206 kali iterasi. Dalam perhitungan ini akan mempercepat dalam proses pendapatan nilai optimal secara ekonomi.



Gambar 4. 7 Perbandingan Uji Validasi

4.2.2. Kasus DED Kedua dengan 6 Pembangkit

Kasus kedua merupakan kasus DED dengan menggunakan rugi-rugi transmisi yang akan dihitung dengan metode perhitungan *losses B-Coefficient*. Kasus kedua akan dihitung menggunakan metode *enhanced lambda* iterasi dengan jumlah pembangkit sebanyak 6 pembangkit yang mensuplai 26 bus interkoneksi. Berikut data-data yang diperlukan untuk perhitungan dalam penerapan *software powergen*.



Gambar 4. *8* *One Line Diagram* pada Kasus 2

Tabel 4. 9 Data Beban

No Bus	Beban		No Bus	Beban	
	MW	Mvar		MW	Mvar
1	51	41	14	24	12
2	22	15	15	70	31
3	64	50	16	55	27
4	25	10	17	78	38
5	50	30	18	153	67

Tabel 4. 9 Data Beban

No Bus	Beban		No Bus	Beban	
	MW	Mvar		MW	Mvar
6	76	29	19	75	15
7	0	0	20	48	27
8	0	0	21	46	23
9	89	50	22	45	22
10	0	0	23	25	12
11	25	15	24	54	27
12	89	48	25	28	13
13	31	15	26	40	20

Tabel 4. 10 Data Generator

No Bus	*Voltage Magnitude*	Generator (MW)	Mvar Limits	
			Min	Max
1	1,025	0	0	0
2	1,020	79	40	250
3	1,025	20	40	150
4	1,050	100	40	80
5	1,045	300	40	160
26	1,015	60	15	50

Tabel 4. 11 Data Kapasitor Shunt

No Bus	Mvar
1	4
4	2
5	5
6	2
11	1,5
12	2
15	0,5
19	5

Tabel 4. 12 Data Tap Transformator

Desain	Setting Tap
2-3	0,960
2-13	0,960

Tabel 4. 12 Data Tap Transformator

Desain	Setting Tap
2-13	1,017
4-8	1,050
4-12	1,050
6-19	0,950
7-9	0,950

Tabel 4. 13 Data Line dan Transformer

No Bus	No Bus	$Rx10^{-3}$ (pu)	$Xx10^{-3}$ (pu)	$1/2B\ x10^{-2}$ (pu)	No Bus	No Bus	$Rx10^{-3}$ (pu)	$Xx10^{-2}$ (pu)	$1/2B\ x10^{-3}$ (pu)
1	2	0,5	4,8	3	10	22	6,9	2,98	5
1	18	1,3	11	6	11	25	96	27	10
2	3	1,4	51,3	5	11	26	16,5	9,7	4
2	7	10,3	58,6	1,8	12	14	32,7	8,02	0
2	8	7,4	32,1	3,9	12	15	18	5,98	0
2	13	3,5	96,7	2,5	13	14	4,6	2,71	1
2	26	32,3	196,7	0	13	15	11,6	6,1	0
3	13	7	5,4	0,05	13	16	17,9	8,88	1
4	8	0,8	24	0,01	14	15	6,9	3,82	0
4	12	1,6	20,7	1,5	15	16	20,9	5,12	0
5	6	6,9	30	9,9	16	17	99	6	0
6	7	5,3	30,6	0,1	16	20	23,9	5,85	0
6	11	9,7	57	0,01	17	18	3,2	6	38
6	18	3,7	22,2	0,12	17	21	229	44,5	0
6	19	3,5	66	4,5	19	23	30	13,1	0
6	21	5	90	2,26	19	24	30	12,5	2
7	8	1,2	6,9	0,01	19	25	119	22,49	4
7	9	0,9	42,9	2,5	20	21	65,7	15,7	0
8	12	2	18	20	20	22	15	3,36	0
9	10	1	49,3	0,1	21	24	47,6	15,1	0
10	12	2,4	13,2	1	22	23	29	9,9	0
10	19	54,7	236	0	22	24	31	8,8	0
10	20	6,6	16	0,1	23	25	98,7	11,6	0

Tabel 4. 14 Data Karakteristik Unit Pembangkit Kasus 2

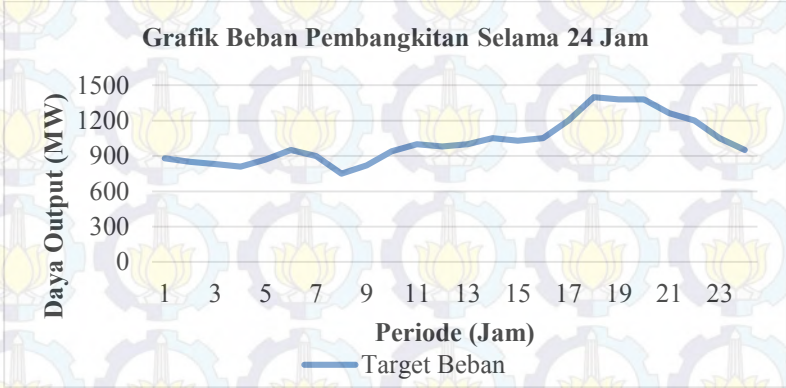
Unit	a	b	c	P_{min}	P_{max}
1	240	7	0.007	100	500
2	200	10	0.0095	50	200
3	220	8,5	0.009	80	300
4	200	11	0,009	50	150
5	220	10,5	0,008	50	200
26	190	12	0,0075	50	120

Data *B-Coefficient* untuk Kasus Kedua

	0,0017	0,0012	0,0007	-0,0001	-0,0005	-0,0002
	0,0012	0,0014	0,0009	0,0001	-0,0006	-0,0001
B :	0,0007	0,0009	0,0031	0,0000	-0,0010	-0,0006
	-0,0001	0,0001	0,0000	0,0024	-0,0006	-0,0008
	-0,0005	-0,0006	-0,0010	-0,0006	0,0129	-0,0002
	-0,0002	-0,0001	-0,0006	-0,0008	-0,0002	0,0150

$B0x10^{-3}$: -0,3908 -0,1297 0,7047 0,0591 0,2161 -0,6635

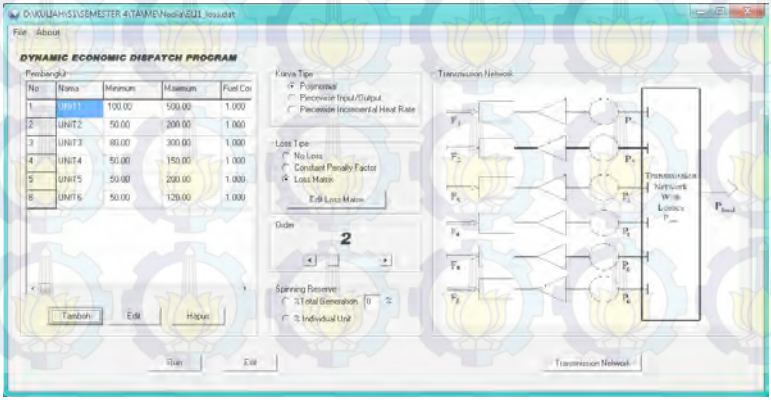
B00 : 0,0056



Gambar 4. 9 Kurva Total Beban kasus 2 selama 24 Periode

Kasus kedua juga dijalankan dengan mengadaptasikan kondisi *real*. Yaitu dengan menggunakan *losses* selama periode waktu 24 jam dengan masing-masing interval selama 1 jam. Dengan menggunakan kurva pembangkitan sesuai dengan kurva total beban pembangkit jawa bali. Beban pembangkitan tergambar dalam kurva gambar 4.7. Untuk melakukan *dynamic economic dispatch* dengan menggunakan *software powergen* dilakukan dengan memasukkan data pada kolom 'Tambah' seperti pada gambar 4.9. Gambar 4.10 menunjukan tampilan menu utama DED pada *software powergen*.

Gambar 4. 10 Form Input Data Pembangkit



Gambar 4. 11 Tampilan Menu Utama DED pada *Powergen*

Seperti pada langkah sebelumnya, dengan memilih tombol '*Run*' maka *software* akan menjalankan program. Selanjutnya pada menu '*Set Up Solution*' inputkan target pembangkitan sesuai pembebanan selama 24 periode. Data pembebanan tercantum pada tabel 4.9.

Tabel 4. 15 Beban Pembangkitan selama 24 Jam Kasus Pertama

Beban	Jam	Beban	Jam
880	1	1000	13
850	2	1050	14
830	3	1030	15
810	4	1050	16
870	5	1200	17
950	6	1400	18
900	7	1380	19
750	8	1380	20
820	9	1263	21
940	10	1200	22
1000	11	1050	23
980	12	950	24

Pada kasus satu ini juga terdapat rugi-rugi transmisi yang menyebakan nilai pembangkitan harus lebih besar dari target. Karena $\sum_{i=1}^{N} Pi = \sum_{i=1}^{N} P_D + Pl$ menunjukan penambahan *losses* pada setiap pembangkit. Dilakukan *running program* menggunkan metode *enhanced lambda* iterasi maka didapat hasil pembagian daya pada generator pada tabel 4.2. Sedangkan *Result* pada *software Powergen* tercantum pada Gambar 4. 12 Perbandingan Daya yang Dibangkitkan dengan Total Beban Target.

Tabel 4. 16 Pembagian Daya Pembangkit pada Kasus 2

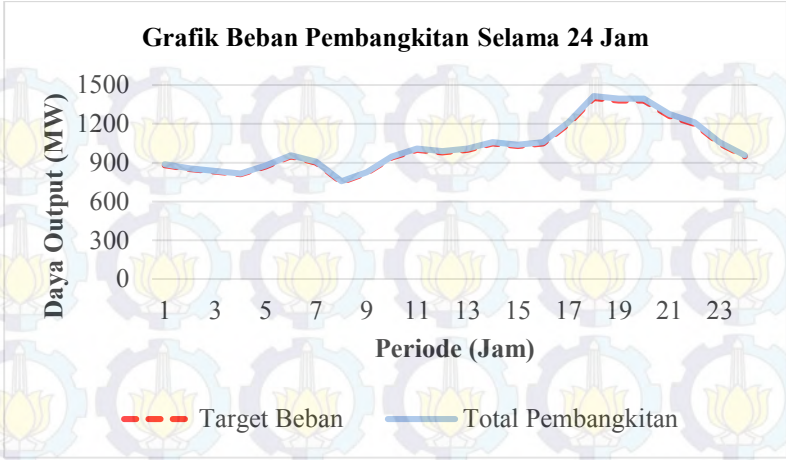
Periode	Unit generator						Generator Cost (Rp/HR)	Load (MW)
	1	2	3	4	5	6		
1	361,93	110,05	197,20	69,55	98,22	50,00	10498.0558	880
2	354,54	104,58	191,50	63,59	92,35	50,00	10131.8105	850
3	349,62	100,94	187,69	59,62	88,44	50,00	9889.4778	830
4	344,70	97,30	183,89	55,66	84,53	50,00	9648.6062	810
5	359,47	108,23	195,30	67,57	96,26	50,00	10375.6071	870
6	379,19	122,82	210,53	83,49	111,88	50,00	11365.4781	950
7	366,86	113,70	201.01	73,53	102,12	50,00	10744.0542	900
8	328,05	84,96	171,02	50,00	71,30	50,00	8935.2149	750
9	347,16	99,12	185,79	57,64	86,48	50,00	9768.8582	820
10	376,73	12,99	208,63	81,49	109,93	50,00	11240.4583	940
11	391,54	131,96	220,06	93,46	121,62	50,00	11996.1054	1000
12	386,60	128,30	216,25	89,47	117,72	50,00	11742.7495	980
13	391,54	131,96	220,06	93,46	121,62	50,00	11996.1054	1000
14	403,44	140,76	229,26	10,09	130,98	51,89	12635.9205	1050
15	398,96	137,45	225,79	99,46	127,46	50,00	12378.9109	1030
16	403,44	140,76	229,26	103,09	130,98	51,89	12635.9205	1050
17	434,44	163,66	253,32	128,39	155,28	76,73	14602.9705	1200
18	479,17	196,76	288,06	150,00	189,96	112,01	17336.0408	1400
19	474,13	193,02	284,14	150,00	186,08	108,09	17056.4626	1380
20	474,13	193,02	284,14	150,00	186,08	108,09	17056.4626	1380

Tabel 4. 16 Pembagian Daya Pembangkit pada Kasus 2

Periode	Unit generator						Generator Cost (Rp/HR)	Load (MW)
	1	2	3	4	5	6		
21	447,50	173,32	263,46	139,07	165,47	87,13	15449.8977	1263
22	434,44	163,66	253,32	128,39	155,28	76,73	14602.9705	1200
23	403,44	140,76	229,26	103,09	130,98	51,89	12635.9205	1050
24	379,19	122,82	210,53	83,49	111,88	50,00	11365.4781	950

Tabel 4. 17 Total Pembangkitan Kasus 2

Periode	P_{losses}	PD	P_{total}	Iterasi	Periode	P_{losses}	PD	P_{total}	Iterasi
1	6,9521	880	886,9520	23	13	8,6449	1000	1008,6449	18
2	6,5685	850	856,5685	19	14	9,4096	1050	1059,4095	21
3	6,3215	830	836,3215	21	15	9,1077	1030	1039,1077	19
4	6,0816	810	816,0814	16	16	9,4096	1050	1059,4095	21
5	6,8225	870	876,8224	22	17	11,8212	1200	1211,8211	29
6	7,9088	950	957,9088	17	18	15,9559	1400	1415,9559	43
7	7,2167	900	907,2166	18	19	15,4570	1380	1395,4569	25
8	5,3217	750	755,3217	16	20	15,4570	1380	1395,4569	25
9	6,2007	820	826,2005	15	21	12,9582	1263	1275,9581	28
10	7,7669	940	947,7669	17	22	11,8212	1200	1211,8211	29
11	8,6449	1000	1008,6449	18	23	9,4096	1050	1059,4095	21
12	8,3452	980	988,3452	16	24	7,9088	950	957,9088	17



Gambar 4. 12 Perbandingan Daya yang Dibangkitkan dengan Total Beban Target

Grafik diatas menunjukan nilai pembangkitan dari generator hampir mendekati nilai targer beban total. Karena adanya penambahan nilai *losses* maka terdapat sedikit perbedaan daya terbangkit dengan daya yang ditargetkan. Sehingga nilai *output*nya berubah. Jumlah dari iterasi menggunakan *enhanced lambda* iterasi lebih singkat jika dibandingkan dengan metode iterasi lambda. Berikut gambar hasil simulasi yang dilakukan pada *software Powergen* dapat dilihat pada

Gambar 4. 12 Perbandingan Daya yang Dibangkitkan dengan Total Beban Target

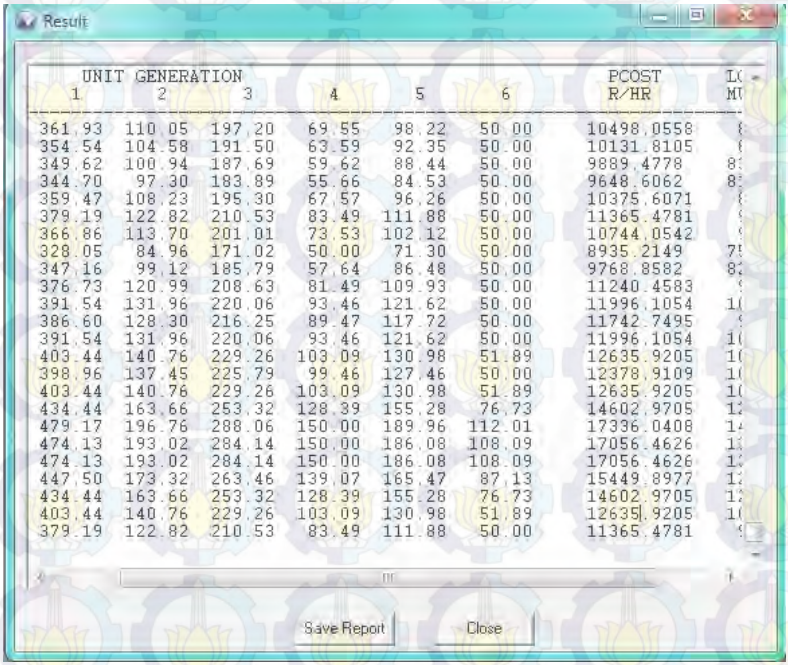
Untuk melakukan validasi hasil simulasi maka dilakukan perbandingan hasil simulasi *powergen* dengan contoh *plan* yang ada pada pada referensi [11]. Tabel 4.13 menunjukan hasil perbandingan antara *Lambda search, Newton-Raphson, Enhanced Lambda* Iterasi.

Tabel 4. 18 Perbandingan Hasil Simulasi Kasus II

Metode / Variabel	Metode			
	Newton-Raphson	Lambda	Enhanced Lambda	PSO
P1 (MW)	719,534	447,50	447,50	447,5370

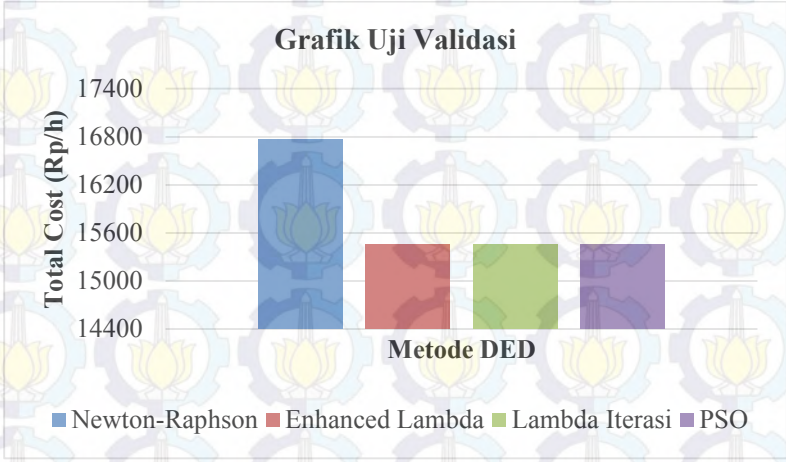
Tabel 4. 18 Perbandingan Hasil Simulasi Kasus II

Metode / Variabel	Metode			
	Newton-Raphson	Lambda	Enhanced Lambda	PSO
P2 (MW)	79,000	173,32	173,32	174,0341
P3 (MW)	20,000	263,46	263,46	263,2430
P4 (MW)	100,000	139,06	139,07	138,8944
P5 (MW)	300,000	165,47	165,47	165,0064
P6 (MW)	60,000	87,13	87,13	87,2395
Ptotal (MW)	1278,534	1275,959	1275,9581	1275,9550
Ploss (MW)	15,53	12,958	12,9582	12,955
Total Cost (Rp/h)	**16760.73**	**15449.9046**	**15449.8977**	**15449.9074**



Gambar 4. 13 Tampilan Hasil Akhir pada *Software Powergen*

Uji validasi dilakukan dengan menggunakan daya pembangkitan sebesar 1263MW. Dengan base MVA sebesar 100MVA. Dari hasil simulasi diatas menunjukan bahwa nilai pembangkitan pada setiap generator hampir sama atau mendekati sama. Biaya pembangkitan *enhanced lambda* iterasi lebih murah jika dibandingkan dengan metode PSO dan lambda Iterasi. Selain itu *enhanced lambda* iterasi mencapai nilai konvergen lebih cepat dengan 28 iterasi. Sedangkan lambda iterasi mencapai nilai lambda yang konvergen selama ±174 kali iterasi. Dalam perhitungan ini akan mempercepat dalam proses pendapatan nilai optimal secara ekonomi. Berikut grafik perbandingan biaya pembangkitan hasil uji validasi tergambar pada gambar Gambar 4. 14 **Perbandingan Uji Validasi**



Gambar 4. 14 Perbandingan Uji Validasi

BAB 5
PENUTUP

5.1. Kesimpulan

Pada tugas akhir ini dalam mengembangkan software powergen berbasis delphi 7.0 maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Metode *enhanced lambda* iterasi juga telah menunjukan bahwa dengan algoritma yang sederhana mampu menyelesaikan permasalahan optimasi *dynamic economic dispatch* dengan mempertimbangkan rugi-rugi transmisi.
2. *Dynamic Economic Dispatch* dengan metode *enhanced* lambda iterasi yang diujikan pada kasus I memiliki jumlah iterasi yang lebih sedikit dibandingkan dengan metode lambda iterasi yaitu sebanyak 35 iterasi. Sedangkan metode lambda iterasi sebanyak 206 iterasi. Pada kasus II juga mendapatkan pembagian daya yang optimal dengan melakukan iterasi sebanyak 19 iterasi menggunakan *enhanced* lambda iterasi dan 174 iterasi menggunakan metode lambda iterasi.
3. Biaya pembangkitan yang dihasilkan dengan menggunakan metode *enhanced* lambda iterasi mendapatkan nilai yang lebih murah dibandingkan metode newton-raphson, lambda iterasi dan PSO. Dimana pada kasus I dan II biaya pembangkitanya memiliki perbedaan sebesar 34 Rp/h dan 1311 Rp/h lebih murah dibandingkan dengan metode newton-raphson. Namun metode ini hanya memiliki perbedaan biaya yang sedikit lebih murah jika dibandingkan dengan metode lambda iterasi dan PSO. Walaupun tidak memiliki perbedaan biaya pembangkitan yang signifikan jika dibandingkan dengan metode tersebut, metode ini telah menunjukan kemajuan yang lebih baik dari metode sebelumnya.

5.2. Saran

1. Untuk aplikasinya dapat diujikan ke dalam sistem yang lebih kompleks lagi seperti sistem jawa bali dengan sistem interkoneksi berbagai jenis pembangkit.
2. Pengambangan ke depannya diharapkan dapat dikembangkan dengan menggunakan batasan *ramp rate, spinning reserve* dan *economic emmision dispatch* yang digabungkan dengan

mempertimbangkan rugi-rugi transmisi sehingga akan semakin mendekati dengan kenyataan sebenarnya di lapangan.

3. Aplikasi yang belum ada di *powergen* adalah perhitungan dengan menggunakan metode perhitungan *economic dispatch* yang digabungkan dengan aplikasi kecerdasan buatan seperti PSO, *qudratic programming* dan lain sebagainya.

DAFTAR PUSTAKA

[1] Penangsang, O., "Analisis Aliran Daya", ITS Press Surabaya, 2012.

[2] Ringlee, R.J. and Williams, D.D. "Economic System Operation Considering Valve Throttling Losses II-Distribution of System Loads by the Method of Dynamic Programming" Power Apparatus and Systems, Part III. Transactions of the American Institute of Electrical Engineers, 1962.

[3] Suryo, H., "Peramalan Beban", ITS Press Surabaya, 20...

[4] Wood Allen J, Wollenberg Bruce F, (1996), "*Power Generation, Operational, and Control*", Second Edition, Jhon Wiley & Sons, Inc.

[5] Jizhong Zhu, "*Optimization of Power System Operation*", IEEE press series on Power engineering, OPSO, John Willey & Sons Inc, America, 2009

[6] Ryzkyanto.H, "Dynamic Economic Dispatch dengan Memperhatikan Ramp-Rate Menggunakan Metode Iterasi Lambda Berbasis Delphi", ITS, 2015.

[7] Dr mas hardi

[8] Kusnassriyanto, "Belajar Pemrograman Delphi", Penerbit Modula Bandung, Bandung, 2011.

[9] Borland Delphi 7, "Developer's Guide", Borland Software Corporation, 2002.

[10] Saadat,H., "Power System Analysis", Mc Graw-Hill, 1999.

[11] Mekhamer.S.F, " A Modified Particle Swarm Optimizer Applied to the Solution of the Economic Dispatch Problem" IEEE press, 2004

---Halaman Ini Sengaja Dikosongkan---

LAMPIRAN A

LEMBAR PENGESAHAN PROPOSAL TUGAS AKHIR

Jurusan Teknik Elektro
Fakultas Teknologi Industri - ITS

TE141599 TUGAS AKHIR – 4 SKS

16 SEP 2015

Nama Mahasiswa	: Nadia Dwiatmo
Nomer Pokok	: 2213 106 039
Bidang Studi	: Teknik Sistem Tenaga
Tugas Diberikan	: Semester Gasal Th. 2015/2016
Dosen Pembimbing	: 1. Prof. Ir. Ontoseno Penangsang M.Sc Ph.D 2. Dedet Candra Riawan,ST., M.Eng., Ph.D.
Judul Tugas Akhir	: **Pengembangan Software Dynamic Economic Dispatch dengan Memperhitungkan Rugi-Rugi Transmisi Menggunakan Metode Enhanced Lambda Iterasi** ***(The Development of Dynamic Economic Dispatch Software with Considering Transmission Losses Using Enhanced Lambda Iteration)***

Uraian Tugas Akhir :

Pembangkit yang effisien adalah pembangkit yang dapat membangkitkan daya sesuai dengan kebutuhan beban namun memiliki harga oprasional yang minimum. Agar tercapai tujuan tersebut maka perlu dilakukan analisa. *Economic dispatch* adalah salah satu analisa yang dilakukan untuk menentukan pembebanan pembangkit secara optimal ekonomi. Pada tugas akhir ini akan dibahas pengembangan software dynamic economic dispatch dengan memperhitungkan rugi-rugi transmisi menggunakan enhanced lambda iterasi. Dimana enhanced lambda iterasi adalah merupakan pengembangan dari lambda iterasi yang tentunya memiliki keunggulan dari metode lambda iterasi. Metode ini merupakan metode klasik yg dapat digunakan pada skala besar. Pada tugas akhir ini juga dipilih menggunakan beban dynamic dengan memperhitungkan rugi-rugi transmisi agar lebih sesuai dengan kondisi real di lapangan. Diharapkan dengan menggunakan metode enhanced lambda iterasi dapat menghasilkan waktu iterasi yang lebih cepat dengan *cost* yang lebih baik. Selain itu software berbasis delphi yang *interface* dapat digunakan sebagai implementasi modul pembelajaran materi kuliah operasi optimum sistem tenaga listrik.

Dosen Pembimbing I,

Prof. Ir. H. Ontoseno Penangsang, M.Sc, Ph.D.
NIP. 194907151974121001

Dosen Pembimbing II,

19/09/15

Dedet Candra Riawan,ST., M.Eng., Ph.D.
Nip. 19731119200031001

Mengetahui,
Jurusan Teknik Elektro FTI - ITS
Ketua,

Dr. Tri Arief Sardjono, ST., MT.
Nip. 197002121995121001

Menyetujui,
Bidang Studi Teknik Sistem Tenaga
Koordinator,

Dr. Ir. Soedibyo, M.MT.
NIP. 195512071980031004

Gambar Lembar Pengesahan Tugas Akhir

---Halaman Ini Sengaja Dikosongkan---

LAMPIRAN B

PEMROGRAMAN POWERGEN BERBASIS DELPHI 7,0

```
unit-Unit 29;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, Menus,unit31, Grids, ExtDlgs, ExtCtrls,
Buttons;

type
  TFrmDEDMain = class(TForm)
    MainMenu1: TMainMenu;
    MnFile: TMenuItem;
    MnNew: TMenuItem;
    MnLoad: TMenuItem;
    N1: TMenuItem;
    MnAbout: TMenuItem;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    RbPoly: TRadioButton;
    RbPIO: TRadioButton;
    RbPINC: TRadioButton;
    GroupBox3: TGroupBox;
    RbNoLoss: TRadioButton;
    RbCONSTPF: TRadioButton;
    RbLossFOrm: TRadioButton;
    Order: TGroupBox;
    ScrOrder: TScrollBar;
    LbOrder: TLabel;
    BtTambah: TButton;
    BTEdit: TButton;
    Button2: TButton;
    Button1: TButton;
    Label1: TLabel;
    ODLoadData: TOpenDialog;
    SgNamaPembangkit: TStringGrid;
    BtHapus: TButton;
    Label2: TLabel;
    SDSaveData: TSaveDialog;
    BtLoss: TButton;
    GroupBox4: TGroupBox;
```

```
    RbIndividu: TRadioButton;
    edtSR: TEdit;
    Label3: TLabel;
    rbSR2: TRadioButton;
    grp1: TGroupBox;
    btn1: TBitBtn;
    img1: TImage;
    dlgOpenPic1: TOpenPictureDialog;
    dlgOpenPic2: TOpenPictureDialog;
    dlgOpenPic3: TOpenPictureDialog;

    procedure Button1Click(Sender: TObject);
    procedure ScrOrderChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure RbPIOClick(Sender: TObject);
    procedure RbPolyClick(Sender: TObject);
    procedure RbPINCClick(Sender: TObject);
    procedure RbNoLossClick(Sender: TObject);
    procedure RbCONSTPFClick(Sender: TObject);
    procedure RbLossFOrmClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure BtTambahClick(Sender: TObject);
    procedure MnLoadClick(Sender: TObject);
    procedure LsNameDblClick(Sender: TObject);
    procedure SgNamaPembangkitDblClick(Sender: TObject);
    procedure BTEditClick(Sender: TObject);
    procedure BtHapusClick(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure MnNewClick(Sender: TObject);
    procedure BtLossClick(Sender: TObject);
    procedure edtSRChange(Sender: TObject);
    procedure RbSRPercentageClick(Sender: TObject);
    procedure RbIndividuClick(Sender: TObject);
    procedure rbSR2Click(Sender: TObject);
    procedure btn1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmDEDMain: TFrmDEDMain;
  tmsg:string;
implementation

uses Unit30, Unit32, Unit34, Unit35, Unit33;

{$R *.dfm}
```

```
function CekGenName(namaGenerator:string):integer;
var s1,s2:string;
  i:integer;
begin
cekGenName:= 0;
s1:=trim(uppercase(namagenerator));
for i:=1 to ngen do
 begin
   s2:=trim(uppercase(genname[i]));
   if s2 = s1 then cekGenname := i;
 end;
end;
Procedure GetNewGeneratorName(var GName:string);
var i: integer;
    s1,s2: string;
begin
   i := 0;
   repeat
     i:=i+1;
     str(i,s1);
     s2 :='UNIT'+TRim(s1);
   until cekGenname(s2) = 0;
   gname:=s2;
end;

procedure TFrmDEDMain.Button1Click(Sender: TObject);
label keluar;
var d,e,f,s:string;
    i,ng,Od:integer;
begin

if RbSR2.Checked then
begin
for i:=1 to ngen do IndividuSR[i]:=0;
end;

if ngen = 0 then   showmessage('Tidak ada pembangkit');
  if ngen>0 then
  begin
    if curvetype =  pio then
    begin
       if not(cekiopoint(ng,od)) then
       begin
          str(od,s);
         showmessage('Error on '+genname[ng]+'  Point '+s);
         goto keluar;
       end;
    end;
    if curvetype =  pinc then
```

```
    begin
       if not(cekihrpoint(ng,od)) then
       begin
          str(od,s);
         showmessage('Error on '+genname[ng]+'  Point '+s);
         goto keluar;
       end;
    end;

    diagflag := true;
    FrmDEDSetupSolution.showmodal;
    if prosesrun then FrmDEDResult.showmodal;
  end;
  keluar:
end;

procedure TFrmDEDMain.ScrOrderChange(Sender: TObject);
var s:string;
    k:integer;
begin
  str(scrorder.Position ,s);
  lborder.Caption := s ;
   curveorder:=scrorder.Position ;
end;

procedure TFrmDEDMain.FormCreate(Sender: TObject);
begin
ngen := 0 ;
losstype :=noloss;
curvetype :=poly;
curveorder:=2;
SgNamaPembangkit.RowCount := 2;
SgNamaPembangkit.ColWidths[0]:=40;
SgNamaPembangkit.cells[0,0]:='No';
SgNamaPembangkit.cells[1,0]:='Nama';
SgNamaPembangkit.cells[2,0]:='Minimum';
SgNamaPembangkit.cells[3,0]:='Maximum';
SgNamaPembangkit.cells[4,0]:='Fuel Cost';
btloss.Enabled :=false;
    filename :='NewDEDFile.dat';


end;


procedure TFrmDEDMain.RbPIOClick(Sender: TObject);
begin
curvetype :=  pio;
end;
```

```
procedure TFrmDEDMain.RbPolyClick(Sender: TObject);
begin
curvetype :=poly;
end;

procedure TFrmDEDMain.RbPINCClick(Sender: TObject);
begin
curvetype :=  PINC;
end;

procedure TFrmDEDMain.RbNoLossClick(Sender: TObject);
var i:integer;
begin
penfacd:=penfac;
for i:=1 to ngen do penfac[i]:=1;

btloss.Enabled :=false;
losstype :=noloss;

end;

procedure TFrmDEDMain.RbCONSTPFClick(Sender: TObject);
begin
penfac :=penfacd;
btloss.Enabled :=true;
btloss.Caption := 'Edit Penalty Factor';
losstype :=constpf;
end;

procedure TFrmDEDMain.RbLossFOrmClick(Sender: TObject);
begin
penfac:=penfacd;
btloss.Enabled :=true;
 btloss.Caption := 'Edit Loss Matrix';
losstype:=lossform;
end;

procedure TFrmDEDMain.Button2Click(Sender: TObject);
begin
close;
end;

procedure TFrmDEDMain.BtTambahClick(Sender: TObject);
var i:integer;
begin
  modedatapembangkit:=1;
  NoGenerator:=ngen+1;
  if NoGenerator>35 then
  begin
    showmessage('Jumlah Maximum Pembangkit 35');
```

```
  end else
  begin
    getNewGeneratorName(genname[NoGenerator]);
    penfac[noGenerator]:=1.0;
    fuelcost[nogenerator]:=1.0;
    frmDEDdataPembangkit.showmodal;
    pgenmax:=0;
    pgenmin:=0;
    for i := 1 to ngen do
       begin
        ihr_ftn( i, pmax[ i ], maxihr[ i ] );
        ihr_ftn( i, pmin[ i ], minihr[ i ] );
        pgenmax := pgenmax + pmax[ i ];
        pgenmin := pgenmin + pmin[ i ];
      end;
    end;
end;

procedure TFrmDEDMain.MnLoadClick(Sender: TObject);
 label keluar;
 var i,k:integer;
 s:string;
begin
if OdLoaddata.Execute then
begin

 filename:= ODLoadData.filename;
 //if not(cekEDCfile(filename )) then
 //begin
 //  showmessage('Bukan File DED ');
 //  goto keluar;
 //end;

 frmDEDmain.caption := filename;

 datainput(filename);
 penfacd:=penfac;

 SgNamaPembangkit.RowCount := ngen+1;
 SgNamaPembangkit.ColWidths[0]:=40;

 for i:=1 to ngen do
 begin
   str(i,s);
   sgnamapembangkit.cells[0,i]:= s;
   sgnamapembangkit.cells[1,i]:= genname[i] ;

   unitmax[i] := pmax[i];
   unitmin[i] := pmin[i];
   str(pmin[i]:7:2,s);
```

```
    sgnamapembangkit.cells[2,i]:= s;
    str(pmax[i]:7:2,s);
    sgnamapembangkit.cells[3,i]:= s;
    str(fuelcost[i]:6:3,s);
    sgnamapembangkit.cells[4,i]:= s;

  end;
  if curvetype =poly then
  begin
   FrmDEDMain.rbpoly.Checked :=true;
  end;
  if curvetype =pinc then
  begin
    FrmDEDMain.rbpinc.Checked :=true;
  end;
  if curvetype =pio then
  begin
    frmDEDMain.rbpio.Checked :=true;
  end;
  if losstype =noloss then frmDEDMain.RbNoLoss.Checked :=true;
  if losstype =lossform then frmDEDMain.RbLossFOrm.Checked :=
true;
  if losstype =constpf then frmDEDMain.RbCONSTPF.Checked
:=true;
  frmDEDMain.scrorder.Position := curveorder;
  frmDEDMain.Caption :=filename;
   keluar:
  end;


end;

procedure TFrmDEDMain.LsNameDblClick(Sender: TObject);
var i:integer;
begin
{  if lsname.Count >0 then
   begin
        NoGenerator:=lsname.ItemIndex+1 ;
        FrmDEDDataPembangkit.showmodal;


   end;}
end;
procedure DUMM;
Begin
{  if lsname.Count >0 then
   begin
        NoGenerator:=lsname.ItemIndex+1 ;
        FrmDEDDataPembangkit.showmodal;
```

```
  end;
end; }

end;

procedure TFrmDEDMain.SgNamaPembangkitDblClick(Sender:
TObject);
begin
modedataPembangkit:=0;
if (sgnamapembangkit.Row >0) and (sgnamapembangkit.row<=ngen
)then
  begin
       NoGenerator:=sgnamapembangkit.Row ;
       FrmDEDDataPembangkit.showmodal;
 end;

end;

procedure TFrmDEDMain.BTEditClick(Sender: TObject);
begin
if ngen = 0 then   showmessage('Tidak ada pembangkit');
if ngen>0 then
begin
modedataPembangkit:=0;
if (sgnamapembangkit.Row >0) and (sgnamapembangkit.row<=ngen
)then
  begin
       NoGenerator:=sgnamapembangkit.Row ;
       FrmDEDDataPembangkit.showmodal;
 end;
 end ;
end;

procedure TFrmDEDMain.BtHapusClick(Sender: TObject);
var n:integer;
   s:string;
   i,j:integer;
begin
if ngen = 0 then   showmessage('Tidak ada pembangkit');

  if ngen>=1 then
  begin
    noGenerator:=sgNamaPembangkit.Row ;
    if (NoGenerator>0) and (NoGenerator<=nGen) Then
    begin
       s:= genname[NoGenerator]+' Dihapus ?';;
       n:= MessageDlg(s, mtConfirmation
        ,[mbOK,mbCancel], 0 );
       if n= mrOk then
```

```
begin
  if Ngen  = 1 then
  begin
     Ngen:= 0;
  end else
  begin
    for i:= NoGenerator to ngen-1 do
    begin
      genname[i]:=genname[i+1];
      penfac[i]:=penfac[i+1];
      pmax[i]:=pmax[i+1];
      pmin[i]:=pmin[i+1];
      fuelcost[i]:=fuelcost[i+1];
      for j:=0 to curveorder do
      begin
        coeff[i,j] :=coeff[i+1,j];
        ihr_mwpoint[i,j]:=ihr_mwpoint[i+1,j];
        ihr_cost[i,j] :=ihr_cost[i+1,j];
        io_mwpoint[i,j]:=io_mwpoint[i+1,j];
        io_cost[i,j]:=io_cost[i+1,j];
      end;
      b0[ i ] :=b0[i+1];
    end;
    for i := NoGenerator to ngen-1 do
    begin
      for j := 1 to ngen  do
      begin
          b[ i, J ]:= b[ i+1, J ];
      end;
    end;
    for i := 1 to ngen-1 do
    begin
      for j := NoGenerator to ngen-1  do
      begin
          b[ i, J ]:= b[ i, J+1 ];
      end;
    end;

    ngen := ngen-1;
    pgenmax:=0;
    pgenmin:=0;
    for i := 1 to ngen do
    begin
       ihr_ftn( i, pmax[ i ], maxihr[ i ] );
       ihr_ftn( i, pmin[ i ], minihr[ i ] );
       pgenmax := pgenmax + pmax[ i ];
       pgenmin := pgenmin + pmin[ i ];
    end;
  end;
  if ngen >1 then
```

```
        begin
            SgNamaPembangkit.RowCount := ngen+1;
        end else
        begin
           SgNamaPembangkit.RowCount := 2;
           sgnamapembangkit.cells[0,1]:= '';
           sgnamapembangkit.cells[1,1]:= '';
           sgnamapembangkit.cells[2,1]:= '';
           sgnamapembangkit.cells[3,1]:= '';
           sgnamapembangkit.cells[4,1]:= '';
        end;
        for i:=1 to ngen do
        begin
             str(i,s);
             sgnamapembangkit.cells[0,i]:= s;
             sgnamapembangkit.cells[1,i]:= genname[i] ;

             str(pmin[i]:7:2,s);
             sgnamapembangkit.cells[2,i]:= s;
             str(pmax[i]:7:2,s);
             sgnamapembangkit.cells[3,i]:= s;
             str(fuelcost[i]:6:3,s);
             sgnamapembangkit.cells[4,i]:= s;
        end;
      end;
   end;
  end;
end;
procedure TFrmDEDMain.N1Click(Sender: TObject);
var d,e,f,s:string;
begin

  getfilename(filename,d,f,e);
  sdsavedata.FileName := f+'_copy' ;
if sdsavedata.Execute then
begin
  s:=trim(sdsavedata.filename);
  getfilename(s,d,f,e);
  if trim (e) = '' then s:=s+'.dat';
  frmDEDMain.Caption :=s;
  dataoutput(s );
end;
end;

procedure TFrmDEDMain.MnNewClick(Sender: TObject);
begin
  ngen:=0;
  SgNamaPembangkit.RowCount := 2;
  sgnamapembangkit.cells[0,1]:= '';
  sgnamapembangkit.cells[1,1]:= '';
```

```
  sgnamapembangkit.cells[2,1]:= '';
  sgnamapembangkit.cells[3,1]:= '';
  sgnamapembangkit.cells[4,1]:= '';
  filename:='Noname.dat'
end;

procedure TFrmDEDMain.BtLossClick(Sender: TObject);
begin
  if ngen>0 then
  begin
    if losstype =constpf then  frmDEDpenfac.showmodal;

    if  losstype=lossform then frmDEDlossform.ShowModal;
  end;
end;

procedure TFrmDEDMain.edtSRChange(Sender: TObject);
var k:integer;
    s:string;
begin
s:=trim(FrmDEDMain.edtSR.Text);
val(s,SR,k);
end;

procedure TFrmDEDMain.RbSRPercentageClick(Sender: TObject);
var i:integer;
begin
edtSR.Enabled := true;
FrmDEDDataPembangkit.edIndividuSR.Enabled:=false;

end;

procedure TFrmDEDMain.RbIndividuClick(Sender: TObject);
begin
edtSR.Enabled := false;
FrmDEDDataPembangkit.edIndividuSR.Enabled:=true;
end;

procedure TFrmDEDMain.rbSR2Click(Sender: TObject);
begin
edtSR.Enabled := true;
FrmDEDDataPembangkit.edIndividuSR.Enabled:=false;
end;

procedure TFrmDEDMain.btn1Click(Sender: TObject);
Var
 stifff:string;

begin
  if ngen = 3 then
```

```
  begin
    stifff:=dlgOpenPic1.FileName;
    img1.Picture.LoadFromFile(stifff);
  end;
  if ngen = 6 then
  begin
    stifff:=dlgOpenPic2.FileName;
    img1.Picture.LoadFromFile(stifff);
  end;
  if ngen = 10 then
  begin
    stifff:=dlgOpenPic3.FileName;
    img1.Picture.LoadFromFile(stifff);
  end;

end;

end.
```

```
unit-Unit 31;

interface
uses  sysutils, Messages, Dialogs;

const

        max_units = 35;
        max_order = 10;
        max_curve_points = 10;
        max_period = 24;
        max_total_segments = 200;
        total_gen_tolerance = 0.0001 ;
        ihr_tolerance = 0.0001 ;
        alpha : real = 0.5; {See note in loss matrix
procedure}

type

        unit_array_real = array[1..max_units] of real;
        unit_name_array = array[1..max_units] of string;
        coefficients = array[0..max_order] of real;
        unit_poly_array = array[1..max_units] of
coefficients;
        curve_points = array[0..max_curve_points] of real;
        unit_curve_array = array[1..max_units] of
curve_points;
        system_ihr_array_real = array[1..max_total_segments]
of real;
        system_ihr_array_integer =
array[1..max_total_segments] of integer;
        B_matrix = array[1..max_units] of unit_array_real;
        filename_array = string;
        curvetype_list = (poly,pinc,pio);
        losstype_list = (noloss,constpf,lossform);
        solution_type_list = (lamsearch,tbllookup,enhanced);
        schedtype_list = (totgen,totload);
        unit_array_period= array[1..max_period] of
unit_array_real;
        period_array_real= array[1..max_period] of real;


var
       ioerr : boolean;
        ioval : integer;
    data_period:unit_array_period;     {Generation each unit
on each period}
     total_period:period_array_real;    {total cost each
period}
```

```
     genname:unit_name_array;            {Generator name
identifier}
     p:unit_array_real;                  {Present value of P}
     pmin:unit_array_real;               {Minimum MW}
     pmax:unit_array_real;               {Maximum MW}
     UR:unit_array_real;
     DR:unit_array_real;
     unitsebelum:unit_array_real;
     ramprate:boolean;
     minihr:unit_array_real;             {Minimum unit
incremental heat rate}
     maxihr:unit_array_real;             {Maximum unit
incremental heat rate}
     fuelcost:unit_array_real;           {Fuel cost ( $/fuel
unit )}
     coeff : unit_poly_array;            {Unit polynomial
coefficients}
     ihr_mwpoint:unit_curve_array;        {MW points on ihr
cost curve}
     ihr_cost:unit_curve_array;           {Cost points for
ihr curve}
     io_mwpoint : unit_curve_array;      {MW Points on unit io
curve}
     io_cost : unit_curve_array;         {Cost points on io
curve}
     minput:unit_array_real;             {Minimum input for
PINC curves }
     penfac:unit_array_real;             {Loss penalty factor}
     penfact:unit_array_real;             {Loss penalty
factor}
     penfacD:unit_array_real;             {Loss penalty
factor}
     unitmin:unit_array_real;             {Initial Minimum
Unit Generation}
     unitmax:unit_array_real;             {Initial Maximum
Unit Generaion}
     totalcost:real;
     loadjam:unit_array_real;

     b00:real;                           {Loss matrix
constant}
     b0:unit_array_real;                 {Loss matrix linear
terms}
     b:B_matrix;                         {Loss matrix
quadratic terms}
     seginccost:system_ihr_array_real;  {Segment inc cost for
table look up }
     segunit:system_ihr_array_integer;  {Unit associated with
segment}
```

```
     segmw:system_ihr_array_real;       {MW contributed by
segment}
     order:system_ihr_array_integer;    {Order routine
output}
     ordvalue:system_ihr_array_real;    {Numbers to be
ordered}
     inputfile:text;
     filename:filename_array;
     title1, title2 : string[80];
     print_output, diagflag, read_data : boolean;
     inputchar,quitflag : char;
     linenumber, ngen : integer;
     mwlosses, lambda : real;
     curvetype_input, losstype_input : string[8];
     number_string : string[20];
     curveorder : integer;
     curvetype : curvetype_list;
     losstype : losstype_list;
     solution_type : solution_type_list;
     schedtype : schedtype_list;
     pgenmax, pgenmin, SRGenTotal : real;
     FF:Text;
     NoGenerator:integer ;
     NomerOrder:integer;
     ModeDataPembangkit:integer;
     ProsesRun:boolean;
     IndividuSR:unit_array_real;
     SR : real;
     numper : integer;

procedure datainput(namafile :string);
procedure ihr_ftn(i : integer; unitmw : real;  var unitihr :
real );
procedure GetFileName(s :string;var d:string;var f:string;var
e :string);
procedure datadump( loadjam:real;unitsebelum:unit_array_real;
var outfile:text );
procedure lambda_search_dispatch( schedmw:real; var lambda :
real);
procedure enhanced_lambda( schedmw:real; var lambda : real);
procedure loss_matrix_ftn;
procedure inverse_ihr_ftn(i : integer; unitihr : real; var
unitmw : real );

procedure order_routine(       numorder : integer;
                              ordertable :
system_ihr_array_real;
                         var   orderindex :
system_ihr_array_integer );
```

```
procedure output_routine( schedmw:real; var outfile : text;
lambda : real );
procedure prod_cost(       i : integer;
                     unitmw : real;
             var   unitcost : real );
procedure dataOutput(namafile :string);
PROCEDURE output_final( VAR  OUTFILE : TEXT );

function cekIoPoint(var unitG,Order:integer):boolean;
function cekIhrPoint(var unitG,Order:integer):boolean;
function CekEdcFile(filename:string): boolean;


implementation
function CekEdcFile(filename:string): boolean;
label keluar;
var s,d,f,e:string;
ff:text;
bc : boolean;
begin
bc:=false;
getfilename(filename,d,f,e);
 s:=trim(uppercase(f))+'.'+uppercase(e);
if s = 'EDC1.DAT' then bc:= true;
if s = 'EDCTEST.DAT' then bc:= true;

if s = 'EDC2.DAT' then bc:= true;
if s = 'EDC3.DAT'  then bc:= true;
if s = 'EDC4.DAT'  then bc:= true;
if s = 'EDC5.DAT'  then bc:= true;

if s = 'EXDED.DAT'  then bc:= true;
if s = 'EX3B.DAT'  then bc:= true;
if s = 'EX3C.DAT'  then bc:= true;
if s = 'EX3D.DAT'  then bc:= true;

if s = 'PR32.DAT'  then bc:= true;
if s = 'PR33.DAT'  then bc:= true;
if s = 'PR38.DAT'   then bc:= true;
if s = 'PR43A.DAT'  then bc:= true;
if s = 'PR43B.DAT'   then bc:= true;
if s = 'EX4D.DAT'   then bc:= true;

if bc = true then goto Keluar;

assign(ff,filename);
reset(ff);
readln(ff,s);
if pos('EDC FILE >>',s)>0 then bc:=true;
close(ff);
```

```
keluar:
cekedcfile:=bc;
end;
function cekIoPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIopoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (io_mwpoint[i,j+1] - io_mwpoint[i,j])<=0 then
      begin
        cekIoPoint:=false;
        UnitG:=i;
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;


function cekIhrPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIhrpoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (ihr_mwpoint[i,j+1] - ihr_mwpoint[i,j])<=0 then
      begin
        cekIhrPoint:=false;
        UnitG:=i;
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;

procedure prod_cost(       i : integer;
                       unitmw : real;
             var   unitcost : real );
```

```
      { Routine to return unit production cost given unit
output in mw}
      { input : unit index = i}
      {         unit MW =  unitmw}
      { output: unit production cost = unitcost}

var
       j : integer;
       partmw, unitihr, segmentcost : real;

label return;

begin

case curvetype of
  poly :                          {Polynomial I/O curve}
      begin
      unitcost := 0;
      for j := curveorder downto 1 do
      unitcost := ( unitcost + coeff[ i,j ] ) * unitmw;

      unitcost := unitcost + coeff[ i,0 ];
      unitcost := unitcost * fuelcost[ i ];
      goto return
      end;

  pinc :                          {Piecewize incremental
curve}
      begin
      unitcost := minput[ i ] * fuelcost[ i ];
      for j := 1 to curveorder do
         begin
         if (unitmw > ihr_mwpoint[ i,j ]) and ( j <
curveorder ) then
                          {Calculate area under complete
segment}
            begin
            segmentcost :=  ( (ihr_cost[i,j] + ihr_cost[i,j-
1] )/2.0 ) *
                              ( ihr_mwpoint[i,j] -
ihr_mwpoint[i,j-1] ) *
                                       fuelcost[ i ];
            unitcost := unitcost + segmentcost;
            end
            else
                          {Calculate area under partial
segment}
            begin
            partmw := (unitmw - ihr_mwpoint[i,j-1] )/
```

```
                              (ihr_mwpoint[i,j] -
ihr_mwpoint[i,j-1] );
            unitihr := ihr_cost[i,j-1] +
                              ( ihr_cost[i,j] - ihr_cost[i,j-1]
) * partmw;
            segmentcost :=  ( (unitihr + ihr_cost[i,j-1])/
2.0 ) *
                              ( unitmw - ihr_mwpoint[i,j-1] )
*
                              fuelcost[ i ];
            unitcost := unitcost + segmentcost;

            goto return;
            end;
         end;

      end;

   pio :                              {Piecewize I/O curve}
      begin
      for j := 1 to curveorder do
         begin
         if io_mwpoint[i,j] > unitmw then
            begin
            partmw := (unitmw-io_mwpoint[i,j-1] ) /
                              (io_mwpoint[i,j]-
io_mwpoint[i,j-1] );
            unitcost := io_cost[i,j-1] +
                              ( io_cost[i,j]-io_cost[i,j-1] )
* partmw;
            unitcost := unitcost * fuelcost[ i ];
            goto return
            end;
         if j = curveorder then     {Unit is at or above
pmax}
            begin
            unitcost := io_cost[ i, j ] * fuelcost[ i ];
            goto return
            end;
         end;
      end;
  end;  { End of case statement}

return:

end; { End procedure }

procedure output_routine( schedmw:real; var outfile : text;
lambda : real );
```

```
var
        limittxt : string[5];
        totalgen, totalload : real;
        unitihr, unitinccost, unitcost : real;
        i : integer;

label return;

begin

writeln(outfile);
writeln(outfile,
     'generator  output   limit  inc cost penalty fact
operating cost');
writeln(outfile,
     '             mw                $/mwhr
$/hr      ');
writeln(outfile,
     '---------  ------   -----  -------- ------------  ----
----------');

totalgen := 0.0;
totalcost := 0.0;

for i := 1 to ngen do
   begin
   write(outfile,genname[i]:9);
   write(outfile, ' ',p[i]:6:4, '    ');
   limittxt := '     ';
   if abs( p[i] - pmin[i] ) < total_gen_tolerance then
limittxt := 'min  ';
   if abs( p[i] - pmax[i] ) < total_gen_tolerance then
limittxt := 'max  ' ;
   write(outfile, limittxt );

   ihr_ftn( i, p[ i ], unitihr) ;    {Get unit incremental
heat rate}

   unitinccost := unitihr * fuelcost[i];
   write(outfile, unitinccost:9:4);
   write(outfile, '   ',penfac[i]:9:4, '    ');

   prod_cost( i, p[ i ], unitcost );          {Calculate unit
operating cost}

   writeln(outfile, '   ',unitcost:9:4);
   totalgen := totalgen + p[i];
   totalcost := totalcost + unitcost;
   end;
writeln(outfile,
```

```
         '---------  ------                                    -
-------------');
write(outfile, ' totals');
write(outfile, totalgen:9:4,
'                                ', totalcost:9:4);
writeln(outfile);
writeln(outfile, ' lambda = ', lambda:10:4 );
writeln(outfile);

if (schedtype = totgen ) and ( losstype <> lossform ) then
goto return;

if schedtype=totload then totalload := schedmw;

if schedtype=totgen then  totalload := totalgen - mwlosses;

writeln(outfile, 'total load = ',totalload:10:4,
                 '  total losses = ',mwlosses:10:4);
if totalload+mwlosses>pgenmax then showmessage('WARNING!!
Load cant be fully served, Load Shedding Needed');
return:

end; { End procedure }

PROCEDURE output_final( VAR  OUTFILE : TEXT );

VAR I,J : INTEGER;

BEGIN


      WRITELN(OUTFILE);

      WRITELN(OUTFILE,'FINAL OUTPUT DYNAMIC ECONOMIC DISPATCH
');
      WRITELN (OUTFILE);
      WRITE(OUTFILE,'PERIOD       UNIT GENERATION ');
      FOR I:=1 TO ngen*8-16 DO WRITE(OUTFILE,' ');
      WRITELN(OUTFILE,' PCOST      LOAD');
      WRITE(OUTFILE,'     ');
      FOR J := 1 TO ngen DO WRITE(OUTFILE,J:8);
      WRITELN(OUTFILE,'          R/HR       MW');
      FOR J := 1 TO 34+ngen*8 DO WRITE(OUTFILE,'-');
      WRITELN(OUTFILE);
      FOR i := 1 TO numper DO
      BEGIN
        WRITE(OUTFILE,i:4,'    ');
        FOR J := 1 TO ngen DO
           WRITE(OUTFILE,' ',data_period[i,j]:6:2,' ');
```

```
        WRITELN(OUTFILE,'    ',total_period[i]:9:4,'
',loadjam[i]:8:1);
        {OPTSTATE := PATH[ OPTSTATE ];   }
      END;

END;


procedure order_routine(        numorder : integer;
                                ordertable :
system_ihr_array_real;
                          var   orderindex :
system_ihr_array_integer );

{ subroutine to order a list, least first
}
{
}
{ input numorder = the number of items to be ordered
}
{ input ordertable = the items to be ordered
}
{ output orderindex = pointer to order value table
}
{
}
{ nxt = Table used in order subroutine
}
{
}
  var
      stop:boolean;
      i,j,top,last,indx:integer;
      nxt : system_ihr_array_integer;

  begin
    for i := 1 to numorder do begin
      if (i <= 1) then begin
        top := 1;
        nxt[ 1 ] := 0;
        end
      else begin
        j := top;
        last := 0;
        repeat
          stop := true;
          if (ordertable[ i ] > ordertable[ j ]) then begin
            last := j;
            j := nxt[ j ];
```

```
            stop  := (j = 0);
            if (stop) then begin
              nxt[ last ] := i;
              nxt[ i ] := 0;
            end
            end
          else begin
            if (j <> top) then begin
              nxt[ last ] := i;                { j not = top }
              nxt[ i ] := j;
              end
            else begin
              top := i;                        { j = top }
              nxt[ i ] := j;
            end;
          end;
        until stop;
      end;
    end;
    indx := 1;
    j := top;
    repeat
      orderindex[ indx ] := j;
      j := nxt[ j ];
      indx := indx + 1;
    until (j = 0);
  end;


procedure inverse_ihr_ftn(       i : integer;
                           unitihr : real;
                       var unitmw : real );

      { Routine to return unit MW given unit incremental heat
rate}
      { input : unit number = i}
      {         unit inc heat rate = unitihr}
      { output: unit MW stored in unitmw}

label  return;

var
        unitihr1, delihr, partihr, dihrdp : real;
        segmentihr : real;
        j, step : integer;

begin

if unitihr >= maxihr[ i ] then
    begin
```

```
    unitmw := pmax[ i ];
    goto return
    end;
if unitihr <= minihr[ i ] then
    begin
    unitmw := pmin[ i ];
    goto return
    end;

case curvetype of
  poly :                          {Polynomial curve}
      begin
      if curveorder <= 1 then
       begin
       if unitihr > coeff[ i,1 ] then unitmw:=pmax[i] else
unitmw:=pmin[i];
       goto return
       end;

      if curveorder = 2 then
       begin
       unitmw := ( unitihr - coeff[ i,1 ] ) / ( 2.0 * coeff[
i,2 ] );
       goto return
       end;

            { for curves of order >= 3 search for unitmw
using Newtons method }

      unitmw := ( pmin[ i ] + pmax[ i ] )/ 2.0;
      step := 0;
      repeat
      step := step + 1;

      unitihr1 := 0;                  {Calc unitihr at unitmw
as unitihr1}
      for j := curveorder downto 2 do
         unitihr1 := ( unitihr1 + j * coeff[i,j] ) * unitmw;
      unitihr1 := unitihr1 + coeff[i,1];
      delihr := unitihr - unitihr1;
      if abs( delihr ) < ihr_tolerance then goto return;

      dihrdp := 0;                    {Calc curve second
derivative}
      for j := curveorder downto 3 do
         dihrdp := ( dihrdp + j*(j-1) * coeff[i,j] ) *
unitmw;
      dihrdp := dihrdp + 2.0 * coeff[ i,2 ];
      unitmw := unitmw + delihr/dihrdp;
```

```
      until step > 35;

      goto return
      end;

  pinc :                              {Piecewize incremental
curve}
      begin
      j := 0 ;
      repeat
      j := j + 1;
      until (ihr_cost[i,j] > unitihr) or
            (j = curveorder);

      partihr := ( unitihr - ihr_cost[i,j-1] )/
                          ( ihr_cost[i,j] - ihr_cost[i,j-1]
);
      unitmw := ihr_mwpoint[i,j-1] +
                    ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] )
* partihr;
      goto return
      end;

   pio :                              {Piecewise I/O curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      if j = curveorder then
         begin
         unitmw := io_mwpoint[i,j];
         goto return
         end;
      segmentihr :=(io_cost[i,j] - io_cost[i,j-1]) /
                              (io_mwpoint[i,j] - io_mwpoint[i,j-
1]);
      until segmentihr >= unitihr;

      unitmw := io_mwpoint[ i,j-1 ];
      goto return
      end;

  end;  { End of case statement}

return:

end; { End procedure }

procedure loss_matrix_ftn;
```

```
      { Routine to calculate losses and penalty factors from
loss formula}
      {  Input:   Table of unit generation p[ i ]}
      {          Loss formula b( i,j ), b0[ i ], b00}
      {  Output:  losses mwlosses and penalty factors penfac[
i ]}
      {  Note: loss formula expects p(i)'s to be in per unit
so divide by 100.}

var
      i, j : integer;
      incloss, penfac_old, penfac_new, inclos : real;

label return;

begin

      mwlosses := b00;
      for i := 1 to ngen do
         begin
         mwlosses := mwlosses + b0[i] *
                        ( p[i]/100.0) + b[i,i] * sqr(
p[i]/100.0 );
         for j := i+1 to ngen do
           mwlosses := mwlosses + 2.0 * ( p[i]/100.0) * (
p[j]/100.0 ) * b[i,j]
         end;

      mwlosses := mwlosses * 100.0;

      for i := 1 to ngen do
         begin
         penfac_old := penfact[ i ];
         incloss := b0[i];
         for j := 1 to ngen do
            incloss := incloss + 2.0 * ( p[j]/100.0 ) *
b[i,j];
         penfac_new := 1.0 / ( 1.0 - incloss );
         penfact[ i ] := penfac_new;
         end;


      for i := 1 to ngen do
         begin
         penfac_old := penfac[ i ];
         incloss := b0[i];
         for j := 1 to ngen do
            incloss := incloss + 2.0 * ( p[j]/100.0 ) *
b[i,j];
         penfac_new := 1.0 / ( 1.0 - incloss );
```

```
          penfac[ i ] := penfac_old + alpha *
                                          (penfac_new -
penfac_old) ;
          penfact[ i ] := penfac_new;
          end;

{ Note, in the formula above the penalty factor is "filtered"
by the }
{ alpha filtering constant. If alpha is set to 1.0 no
filtering action }
{ takes place, if alpha is 0.0 penfac is constant at 1.0 ,
suggested }
{ value for alpha is 0.5 to 0.9 }

return:

end; { End procedure }


procedure lambda_search_dispatch( schedmw:real; var lambda :
real);

var
          i, n, lossiter : integer;
          lambdamin, lambdamax : real;
          lambdastart, deltalambda, targetgen : real;
          unitihr, unitmw, totalgen :  real;
          endloop : boolean;

begin

for i := 1 to ngen do                    { Set unit output to
midrange}
  begin
  p[ i ] := ( pmin[ i ] + pmax[ i ] ) / 2.0
  end;

lossiter := 0;
endloop := false;

repeat                                   {Top of iterative
loop with losses}

  lambdamin := 10000.0;
  lambdamax := 0.0;
  mwlosses := 0 ;
  if losstype = lossform then   { Calc losses and pen
factors}
  begin
    loss_matrix_ftn;
```

```
    writeln(ff);
    writeln(ff,' mw losses = ',mwlosses:10:1);
  end;

  for i := 1 to ngen do                    {Calculate max and
min lambdas}
    begin
    ihr_ftn (i,pmax[i],maxihr[i]);
    lambda := maxihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda > lambdamax then lambdamax := lambda;
    ihr_ftn (i,pmin[i],minihr[i]);
    lambda := minihr[ i ] * penfac[ i ] * fuelcost[ i ];
    if lambda < lambdamin then lambdamin := lambda
    end;



    writeln(ff,' lambda limits =
',lambdamin:10:4,lambdamax:10:4);


  lambdastart := ( lambdamax + lambdamin ) / 2.0;
  deltalambda := ( lambdamax - lambdamin ) / 2.0;


   writeln(ff,' lambdastart deltalambda =
',lambdastart:10:4,deltalambda:10:4);
  {Set up total generation target}
  if schedtype = totgen then targetgen := schedmw;
  if schedtype = totload then targetgen := schedmw +
mwlosses;
  {Lambda search}
  lambda := lambdastart;
  writeln(ff, ' targetgen = ',targetgen:10:1);

  n := 0;
  repeat                              {Top of lambda search
loop}
      n := n + 1;
      totalgen := 0;
      for i := 1 to ngen do
         begin
         unitihr := lambda / ( penfac[ i ] * fuelcost[ i ] )
;
         inverse_ihr_ftn( i, unitihr, unitmw );  {For given
unitihr get unitmw}
         p[ i ] := unitmw;
         totalgen := totalgen + p[ i ]
         end;
```

```
          writeln(ff,' lambda = ',lambda:2:4,'  totalgen =
',totalgen:10:3,' delta lambda = ',deltalambda:1:4);


      if abs( totalgen - targetgen ) >= total_gen_tolerance
then
        begin
        if totalgen > targetgen then lambda := lambda -
deltalambda;
        if totalgen < targetgen then lambda := lambda +
deltalambda;
        deltalambda := deltalambda / 2.0
        end;

      until ( abs( totalgen - targetgen ) <
total_gen_tolerance ) or
            ( n > 35 ) ;

{See if another loss iteration is needed}

      if losstype <> lossform then endloop := true;
      lossiter := lossiter + 1;
      if lossiter > 10 then endloop := true ;
      if  abs( totalgen - targetgen ) > total_gen_tolerance
then  ProsesRun:=false;
      //else ProsesRun:=True;        //check if the program
really get the right result

until endloop;


end; { End Lambda search procedure }

procedure Enhanced_lambda (schedmw:real; var lambda : real);
var
        i, n, lossiter,iterasi : integer;
        lambdamin, lambdamax : real;
        lambdastart, deltalambda, targetgen, targetgen1:
real;
        unitihr,a,b, unitmw, totalgen, eli1, eli2, eli3,
eli4 : real;
        penfaclama,penfacbaru,lambda1, lambda2, lambda3,
ptot, ptot1, ptot2: real;
        deltap,deltap1,deltap2,deltap3 : real;
        endloop : boolean;

begin
```

```
  if schedtype = totgen then targetgen := schedmw;
  if schedtype = totload then targetgen := schedmw;// +
mwlosses;

  iterasi := 1;
  repeat
    if iterasi = 1 then
      begin
        eli1:=0;
        eli2:=0;
        ptot:=0;
        for i:=1 to ngen do
          begin
            eli1:= eli1 +  1 / (2* coeff[i,2]);
            eli2:= eli2 + coeff[i,1] / (2* coeff[i,2]);
          end;
        lambda1 := (targetgen + eli2) / eli1;

        for i:=1 to ngen do
          begin
            p[i]:= (lambda1 - coeff[i,1]) / (2 * coeff[i,2]);
{hitung p}
            if p[i] <= pmin[i] then p[i] := pmin[i];
{check limit nilai pmin}
            if p[i] >= pmax[i] then p[i] := pmax[i];
{check limit nilai pmax}
            Ptot:= ptot + p[i];
{hitung ptot}

          end;
        Writeln( ff, 'iteration= 1 ', iterasi);

        lambda:=lambda1;

      end;

    if iterasi = 2 then
      begin
        ptot1:=0;
        lambdamax := lambda1 * 1.1;
        if deltap1 > 0.0001 then lambda2:=lambdamax;
        lambdamin := lambda1 * 0.9;
        if deltap1 < 0.0001 then lambda2:=lambdamin;

        for i:=1 to ngen do
          begin
            p[i]:= (lambda2 - coeff[i,1]) / (2 * coeff[i,2]);
{hitung p}
```

```
            if p[i] <= pmin[i] then p[i] := pmin[i];
{check limit nilai pmin}
            if p[i] >= pmax[i] then p[i] := pmax[i];
{check limit nilai pmax}
            Ptot1:= ptot1 + p[i];
{hitung ptot}

          end;

        Writeln( ff, 'iteration=2 ', iterasi);

        lambda:=lambda2;

      end;

    if iterasi >=3 then
      begin
        ptot2 := 0;
          repeat
            eli3 := 0;
            eli4 := 0;
            a := lambda2 * deltap1;
            b := lambda1 * deltap2;
            eli3 := a - b;
            eli4 := deltap1 - deltap2;
            if eli4=0 then eli4:=deltap1+deltap2;
            lambda3 := eli3 / eli4;
            for i:=1 to ngen do
               begin
                  p[i]:= (lambda3 - coeff[i,1])/(2 *
coeff[i,2]);       {hitung p}
                  if p[i] <= pmin[i] then p[i] := pmin[i];
{check limit nilai pmin}
                  if p[i] >= pmax[i] then p[i] := pmax[i];
{check limit nilai pmax}
                  ptot2:= ptot2 + p[i];
{hitung ptot}
               end;
            deltap3 := targetgen - ptot2;
{hitung delta p}
            Writeln( ff, 'iteration= ', iterasi);

            lambda1:=lambda2;
            lambda2:=lambda3;
            deltap := deltap3;
            deltap1:=deltap2;
            deltap2:=deltap3;
            ptot2 :=0;
            iterasi := iterasi+1;
```

```
          until (deltap >= -0.0002) and (deltap < 0.0002);//
or (iterasi >= 5);
      end;

  iterasi := iterasi + 1;
  until (deltap >= -0.0002) and (deltap < 0.0002);// or
(iterasi >= 5);

 if losstype = lossform then
 begin
 if losstype = lossform then   { Calc losses and pen factors}
  begin
  lossiter := 0;
  iterasi := 1;
  targetgen1:=targetgen;

    mwlosses := 0 ;
    for i:= 1 to ngen do
    if losstype = lossform then   { Calc losses and pen
factors}
      begin
        loss_matrix_ftn;
      end;

    targetgen := targetgen1 +  mwlosses;

    repeat
    if iterasi = 1 then
      begin
        eli1:=0;
        eli2:=0;
        ptot:=0;
        for i:=1 to ngen do
          begin
            eli1:= eli1 +  1 / (2* coeff[i,2]);
            eli2:= eli2 + coeff[i,1] / (2* coeff[i,2]);
          end;
        lambda1 := (targetgen + eli2) / eli1;


        for i:=1 to ngen do
          begin
            p[i]:= (lambda1 - coeff[i,1]*penfact[i]) / (2 *
coeff[i,2]*penfact[i]);       {hitung p}
            if p[i] <= pmin[i] then p[i] := pmin[i];
{check limit nilai pmin}
            if p[i] >= pmax[i] then p[i] := pmax[i];
{check limit nilai pmax}
            Ptot:= ptot + p[i];
{hitung ptot}
```

```
          end;
        deltap1 := targetgen - ptot;
{hitung delta p}
        Writeln( ff, 'iteration= ', iterasi);

        deltap := deltap1;


        mwlosses := 0 ;
        for i:= 1 to ngen do
        if losstype = lossform then   { Calc losses and pen
factors}
        begin
          loss_matrix_ftn;
        end;

       targetgen := targetgen1 +  mwlosses;
      end;

    if iterasi = 2 then
      begin
        ptot1:=0;
        lambdamax := lambda1 * 1.1;
        if deltap1 > 0.0001 then lambda2:=lambdamax;
        lambdamin := lambda1 * 0.9;
        if deltap1 < 0.0001 then lambda2:=lambdamin;

        for i:=1 to ngen do
          begin
            p[i]:= (lambda2 - coeff[i,1]*penfact[i]) / (2 *
coeff[i,2]*penfact[i]);       {hitung p}
            if p[i] <= pmin[i] then p[i] := pmin[i];
{check limit nilai pmin}
            if p[i] >= pmax[i] then p[i] := pmax[i];
{check limit nilai pmax}
              Ptot1:= ptot1 + p[i];
{hitung ptot}


          end;
        deltap2 := targetgen - ptot1;
{hitung delta p}

        deltap := deltap2;


        mwlosses := 0 ;
        for i:= 1 to ngen do
```

```
        if losstype = lossform then   { Calc losses and pen
factors}
        begin
          loss_matrix_ftn;
        end;

      targetgen := targetgen1 +  mwlosses;

    end;

   if iterasi >=3 then
     begin
       ptot2 := 0;
         repeat

             eli3 := 0;
             eli4 := 0;
             a := lambda2 * deltap1;
             b := lambda1 * deltap2;
             eli3 := a - b;
             eli4 := deltap1 - deltap2;
             lambda3 := eli3 / eli4;
             for i:=1 to ngen do
                begin
                   p[i]:= (lambda3 -
coeff[i,1]*penfact[i])/(2 * coeff[i,2]*penfact[i]);
{hitung p}
                   if p[i] <= pmin[i] then p[i] := pmin[i];
{check limit nilai pmin}
                   if p[i] >= pmax[i] then p[i] := pmax[i];
{check limit nilai pmax}
                   ptot2:= ptot2 + p[i];

                end;
             deltap3 := targetgen - ptot2;
{hitung delta p}
             Writeln( ff, 'iteration= ', iterasi);

             lambda1:=lambda2;
             lambda2:= lambda3;
             deltap := deltap3;
             deltap1:= deltap2;
             deltap2:= deltap3;
             ptot2 :=0;
             iterasi := iterasi+1;

             mwlosses := 0 ;
             for i:= 1 to ngen do
             if losstype = lossform then   { Calc losses and
pen factors}
```

```
            begin
              loss_matrix_ftn;
            end;

            targetgen := targetgen1 +  mwlosses;
          until (deltap >= -0.00002) and (deltap < 0.0002);//
or (iterasi > 5);
      end;

     iterasi := iterasi + 1;
   until (deltap >= -0.00002) and (deltap < 0.0002) or
(iterasi > 5);
   end;
   end;
   end; { End Enhanced Lambda Iteration procedure }



procedure GetFileName(s :string;var d:string;var f:string;var
e :string);
var ss ,sd:string;
  n:integer;
begin
  ss := s;
  d := '';
  n :=  pos('\',ss);
  while  n>0 do
  begin
    sd := copy(ss,1,n);
    d := d+sd;
    delete(ss,1,n);
  n :=  pos('\',ss);
  end;
  n := pos('.',ss);
  if n = 0 then
  begin
     f := ss;
     e:= '';
  end else
  begin
     f:=copy(ss,1,n-1);
     delete(ss,1,n);
     e := ss;
  end;
end;
procedure IOCheck( linenumber : integer );
begin
  IOVal := IOresult;
```

```
  IOErr := (IOVal <> 0);
end; { of proc IOCheck }


procedure datainput(namafile :string);

label quit;

var i,j,k,jj : integer;
    a : char;

begin
print_output := True;

linenumber := 0;

assign(inputfile, namafile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                  }
{ Open data file   }
{                  }
reset(inputfile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                  }
{ Read file header }
{                  }
linenumber := linenumber + 1;
readln( inputfile, title1 );
     iocheck( linenumber );
     if ioerr then goto quit;
linenumber := linenumber + 1;
readln( inputfile, title2 );
     iocheck( linenumber );
     if ioerr then goto quit;

     {                                                  }
     {  Read Number of generators, curve type, loss type  }
     {                                                  }
linenumber := linenumber + 1;
read( inputfile, ngen );
     iocheck( linenumber );
     if ioerr then goto quit;

repeat
read( inputfile, inputchar );
        iocheck( linenumber );
        if ioerr then goto quit;
until inputchar <> ' ';
```

```
curvetype_input := inputchar;
repeat
read( inputfile, inputchar );
      iocheck( linenumber );
      if ioerr then goto quit;
if inputchar <>' ' then curvetype_input := curvetype_input +
inputchar;
until inputchar = ' ';

read( inputfile, curveorder );
      iocheck( linenumber );
      if ioerr then goto quit;

repeat
read(inputfile, inputchar );
     iocheck( linenumber );
     if ioerr then goto quit;
until inputchar <> ' ';
losstype_input := inputchar;
readln(inputfile);

 {                                                          }
 {  Set up internal variables for curvetype and losstype   }
 {                                                          }

if (curvetype_input = 'poly') or
   (curvetype_input = 'POLY') then curvetype := poly;

if (curvetype_input = 'pinc') or
   (curvetype_input = 'PINC') then curvetype := pinc;

if (curvetype_input = 'pio' ) or
   (curvetype_input = 'PIO' ) then curvetype := pio;

if (losstype_input = 'n'  ) or
   (losstype_input = 'N'  ) then losstype := noloss;

if (losstype_input = 'c'  ) or
   (losstype_input = 'C'  ) then losstype := constpf;

if (losstype_input = 'l'  ) or
   (losstype_input = 'L'  ) then losstype := lossform;


        {                    }
        { Read generator data }
        {                    }

for i := 1 to ngen do
   begin  { Read generator name }
```

```
      linenumber := linenumber + 1;
      repeat
      read( inputfile, inputchar );
            iocheck( linenumber );
            if ioerr then goto quit;
      until inputchar <> ' ';
      genname[i] := inputchar;
      repeat
      read( inputfile, inputchar );
            iocheck( linenumber );
            if ioerr then goto quit;
      if inputchar <>' ' then genname[i] := genname[i] +
inputchar;
      until inputchar = ' ';
         {                                        }
         { Read generator min, max, fuelcost }
         {                                        }
      readln( inputfile, pmin[i], pmax[i], fuelcost[i],
DR[i], UR[i], unitsebelum[i] );
            iocheck( linenumber );
            if ioerr then goto quit;
         {                                        }
         { Read generator cost curve data    }
         {                                        }
      case curvetype of
        poly :
              begin  { read polynomial curve data}
              for j := 0 to curveorder do
                begin
                  linenumber := linenumber + 1;
                  readln( inputfile, coeff[i,j] );
                   iocheck( linenumber );
                   if ioerr then goto quit;
                end;
              end;

        pinc :
              begin  { read piecewise incremental cost curve
data}
              linenumber := linenumber + 1;
              readln( inputfile, minput[i] );
                iocheck( linenumber );
                if ioerr then goto quit;
              for j := 0 to curveorder  do
                begin
                  linenumber := linenumber + 1;
                  readln( inputfile, ihr_mwpoint[i,j],
ihr_cost[i,j] );
                    iocheck( linenumber );
                    if ioerr then goto quit;
```

```
                end;
              end;

        pio  :
              begin  { read piecewise I/O curve data}
              for j := 0 to curveorder do
                begin
                  linenumber := linenumber + 1;
                  readln( inputfile, io_mwpoint[i,j],
io_cost[i,j] );
                   iocheck( linenumber );
                   if ioerr then goto quit;
                end;
              end;

        end;  { End of case statement}
    end;
        {                  }
        { Read loss data }
        {                  }

case losstype of

  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0
        end;

  constpf  :
      begin   { read constant penalty factor data}
        linenumber := linenumber + 1;
        for i := 1 to  ngen do
         begin
         if i < ngen then
         read( inputfile, penfac[i] )
         else
         readln( inputfile, penfac[ngen] );
            iocheck( linenumber );
            if ioerr then goto quit
         end;
      end;

  lossform :

      begin  { read loss formula data}

      linenumber := linenumber + 1;
      readln( inputfile, b00 );
            iocheck( linenumber );
```

```
            if ioerr then goto quit;

        linenumber := linenumber + 1;
        for j := 1 to  ngen  do
          begin
          if j < ngen then
             read( inputfile, b0[ j ] )
             else
             readln( inputfile, b0[ngen] );
              iocheck( linenumber );
              if ioerr then goto quit;
        end;

        for i := 1 to ngen do
          begin
          linenumber := linenumber + 1;
          for j := 1 to ngen  do
           begin
           if j < ngen then
              read( inputfile, b[ i, j ] )
              else
              readln( inputfile, b[ i, ngen ] );
              iocheck( linenumber );
              if ioerr then goto quit
           end;
          end;
          for i := 1 to ngen do       { Init penalty factors}
                penfac[ i ] := 1.0
        end;
    end;  { End of case statement}
{                                     }
{  End of data input, close file      }
{                                     }

close ( inputfile );

{A very useful table is the incremental cost at the max and
min of each unit. }
{These are calculated and stored in tables maxihr and
minihr.}
{Also calculate the max and min generation}



quit:

end;   { end of data input }
```

```
procedure dataOutput(namafile :string);
var i,j,k,jj : integer;
    a : char;
    d,e,f:string;
   var inputfile:text;
begin
getfilename(namafile,d,f,e);
assign(inputfile, namafile);
rewrite(inputfile);
writeLn( inputfile, 'EDC FILE >> '+f+'.'+e );
writeln( inputfile,
'=====================================');
write( inputfile, ngen );
write(inputfile,' ');

if  curvetype = poly then write(inputfile,'POLY');
if  curvetype = pinc then write(inputfile,'PINC');
if  curvetype = PIO then write(inputfile,'PIO');
write(inputfile,' ');
write(inputfile,curveorder);
write(inputfile,' ');

if losstype = noloss  then write(inputfile,'NOLOSS');
if losstype = constpf  then write(inputfile,'CONSTPF');
if losstype = lossform  then write(inputfile,'LOSSFORM');
writeln(inputfile);
for i := 1 to ngen do
   begin
      write(inputfile,genname[i]);
      write(inputfile,' ');
      writeln( inputfile, pmin[i]:15:6, pmax[i]:15:6,
fuelcost[i]:15:6, DR[i]:15:6, UR[i]:15:6, unitsebelum[i]:15:6
);
    case curvetype of
        poly :
              begin
              for j := 0 to curveorder do
                begin
                  writeln( inputfile, coeff[i,j]:15:6 );
                end;
              end;

        pinc :
              begin
              writeln( inputfile, minput[i]:15:6 );
              for j := 0 to curveorder  do
                begin
```

```
                    writeln( inputfile, ihr_mwpoint[i,j]:15:6,
ihr_cost[i,j]:15:6 );
                  end;
                end;

        pio  :
                begin
                for j := 0 to curveorder do
                  begin
                    writeln( inputfile, io_mwpoint[i,j]:15:6,
io_cost[i,j]:15:6);
                  end;
                end;

        end;
    end;

case losstype of

  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0
        end;

  constpf  :
      begin
        for i := 1 to  ngen do
        begin
        if i < ngen then
        begin
        write( inputfile, penfac[i]);
        write(inputfile,' ');

        end
        else writeln( inputfile, penfac[ngen]);
        end;
      end;

  lossform :

      begin

      linenumber := linenumber + 1;
      writeln( inputfile, b00);
       for j := 1 to  ngen  do
        begin
        if j < ngen then
           begin
           write( inputfile, b0[ j ]);
```

```
            write(inputfile,' ');
            end
            else
            writeln( inputfile, b0[ngen]);
          end;

        for i := 1 to ngen do
          begin
          for j := 1 to ngen  do
           begin
           if j < ngen then
             begin
             write( inputfile, b[i, j]);
             write(inputfile,' ');

             end
             else
             writeln( inputfile, b[i,ngen]);
           end;
          end;

          for i := 1 to ngen do      { Init penalty factors}
                penfac[ i ] := 1.0

      end;

   end;


close ( inputfile );



end;

procedure ihr_ftn(          i : integer;
                        unitmw : real;
                    var unitihr : real );

      { Routine to return unit incremental heat rate given
unit output in MW }
      { input : unit index = i}
      {         unit mw = unitmw}
      { output: unit incremental heat rate = unitihr}

var
         partmw : real;
         j : integer;

begin
```

```
case curvetype of
  poly :                        {Polynomial I/O curve}
      begin
      unitihr := 0.0;
      for j := curveorder downto 2 do
        unitihr := ( unitihr + j * coeff[ i,j ] ) * unitmw;
      unitihr := unitihr + coeff[ i,1 ]
      end;

  pinc :                        {Piecewize incremental curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      until (ihr_mwpoint[ i,j ] > unitmw) or (j =
curveorder);

      partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                        (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1]
);
      unitihr := ihr_cost[i,j-1] + ( ihr_cost[i,j] -
ihr_cost[i,j-1] ) * partmw
      end;

  pio  :                        {Piecewize I/O curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      until (io_mwpoint[ i,j ] > unitmw) or (j = curveorder);

      unitihr := (io_cost[i,j] - io_cost[i,j-1] ) /
                          ( io_mwpoint[i,j] - io_mwpoint[i,j-
1] )
      end;
  end;  { End of case statement}


end; { End ihr_ftn procedure }

procedure datadump( loadjam:real; unitsebelum:unit_array_real
;var outfile:text );


var
   i,j: integer;

begin
```

```
for i := 1 to ngen do
begin
pmax[i]:=unitmax[i]-(individuSR[i]/100)*unitmax[i];
pmin[i]:=unitmin[i];
if unitsebelum[i] <> 0 then if ramprate then
  begin
  if (unitsebelum[i]+UR[i])<(unitmax[i]-
(IndividuSR[i]/100)*unitmax[i]) then pmax[i] :=
unitsebelum[i]+UR[i];
  if (unitsebelum[i]-DR[i])>(unitmin[i]) then pmin[i] :=
unitsebelum[i]-DR[i];
  end;
end;

writeln(outfile);
writeln(outfile, title1);
writeln(outfile, title2);
writeln(outfile);
writeln(outfile ,' number of generator units = ',ngen );

case curvetype of
  poly : writeln(outfile ,' unit curve type = poly ');
  pinc : writeln(outfile ,' unit curve type = pinc ');
  pio  : writeln(outfile ,' unit curve type = pio');
  end;  { End of case statement}

writeln(outfile ,' curve order = ',curveorder);

case losstype  of
  noloss :   writeln(outfile ,' network loss representation =
noloss ');
  constpf :  writeln(outfile ,' network loss representation =
constpf ');
  lossform : writeln(outfile ,' network loss representation =
lossform ');
  end;  { End of case statement}
for i := 1 to ngen do
  begin
   writeln(outfile);
   write(outfile, genname[i],' limits = ',pmin[i]:7:2, '
',pmax[i]:7:2 );
   writeln(outfile, ' fuelcost = ',fuelcost[i]:10:4 );
     case curvetype of
          poly :
              begin
              writeln(outfile,'  polynomial coefficients' );
              for j := 0 to curveorder do
                begin
                writeln(outfile, coeff[i,j]:15:6);
                end;
```

```
            end;
        pinc :
            begin
            writeln(outfile,' incremental cost curve
points');
            writeln(outfile,'input at pmin =
',minput[i]:10:2);
            for j := 0 to curveorder do
              begin

writeln(outfile,ihr_mwpoint[i,j]:9:2,ihr_cost[i,j]:9:3 )
              end;
            writeln(outfile);
            end;
        pio :
            begin
            writeln(outfile,' cost curve points');
            for j := 0 to curveorder do
              begin
              writeln(outfile,io_mwpoint[i,j]:9:2,'
',io_cost[i,j]:9:3 )
              end;
            writeln(outfile);
            end;
        end; {end of case statement }
  end;
case losstype  of

  constpf :
              begin
              writeln(outfile);
              writeln(outfile,' Penalty Factors');
              for i := 1 to ngen do
                begin
                writeln(outfile,' penalty factor ',i,'
',penfac[i]:10:3)
                end;
              writeln(outfile);
              end;

  lossform :
              begin
              writeln(outfile);
              writeln(outfile,' Loss Formula');
              writeln(outfile,'B00 = ',b00:10:4);
              writeln(outfile);
              writeln(outfile,'B0 = ');
              for i := 1 to ngen do
                 if i < ngen then
                 write(outfile,b0[i]:10:4,' ')
```

```
                    else
                    writeln(outfile,b0[i]:10:4);
                writeln(outfile);
                writeln(outfile,' B = ');
                for i := 1 to ngen do
                   begin
                        for j := 1 to ngen do
                           if j < ngen then
                           write(outfile,b[i,j]:10:4,' ')
                           else
                           writeln(outfile,b[i,ngen]:10:4);
                   end;
                writeln(outfile)
                end;
  end;  { End of case statement}
    if solution_type = lamsearch then writeln(outfile,'using
lambda search');
    if solution_type = enhanced then writeln(outfile,'using
Enhanced Lambda Iteration');
                                  // else
writeln(outfile,'using tablelookup');
      writeln(outfile);
      {if schedtype = totgen then
         writeln(outfile, ' total generation schedule =
',schedmw:10:1)
      else
         writeln(outfile, ' total load schedule = ',
schedmw:10:1);
      if losstype = lossform then writeln(outfile, ' using
loss formula ')
                             else writeln(outfile,' losses
neglected');
      writeln(outfile); }
end; { End procedure }


end.
```

```
unit-Unit 32;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls,unit31, Grids;

type
  TFrmDEDSetUpSolution = class(TForm)
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    Lb: TLabel;
    LbMaxGe: TLabel;
    LbMinG: TLabel;
    Button1: TButton;
    SgLoad: TStringGrid;
    Label4: TLabel;
    Ednuml: TEdit;
    LbSpinningReserve: TLabel;
    GroupBox4: TGroupBox;
    RbRampYes: TRadioButton;
    RbRampNo: TRadioButton;
    rbELI: TRadioButton;
    rblamsearch: TRadioButton;


    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure EdnumlChange(Sender: TObject);

    procedure SgLoadKeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormClose(Sender: TObject; var Action:
TCloseAction);
    procedure rblamsearchClick(Sender: TObject);
    procedure rbELIClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FrmDEDSetUpSolution: TFrmDEDSetUpSolution;
  sgLoad : TstringGrid;
```

```
implementation

uses Unit33, Unit29;

{$R *.dfm}


procedure TFrmDEDSetUpSolution.FormActivate(Sender: TObject);
var s:string;
    k,i:integer;
begin

solution_type  := lamsearch;
schedtype := totload;
     pgenmax := 0.0;
pgenmin := 0.0;
SRgenTotal:=0.0;

for i := 1 to ngen do
  begin
    ihr_ftn( i, pmax[ i ], maxihr[ i ] );
    ihr_ftn( i, pmin[ i ], minihr[ i ] );

{ calculate maximum and minimum generation available from
generators }
    SRGenTotal := SRgenTotal + ((IndividuSR[i]/100)*pmax[i]);
    pmax[i] := pmax[i] - ((IndividuSR[i]/100)*pmax[i]);
    pgenmax := pgenmax + pmax[ i ];
    pgenmin := pgenmin + pmin[ i ];

  end;

if FrmDEDMain.rbSR2.Checked then
  begin
    SRGenTotal := pgenmax*SR/100;
    pgenmax := pgenmax-SRGenTotal;
  end;

prosesrun:=false;
str(pgenmax:10:1,s);
lbMaxGe.caption:=' Maximum generation is :'+s;
str(pgenmin:10:1,s);
lbming.caption:= ' Minimum generation is :'+s;
str (SRGenTotal:10:1,s);
LbSpinningReserve.caption := ' Spinning Reserve is : '+s;
LbSpinningReserve.Visible:=true;
if SRGenTotal =0 then LbSpinningReserve.Visible:=false;
end;

procedure PRoses;
```

```
var a,s,e,f,d:string;
 cekiter,k,i:integer;
begin
  if FrmDEDSetUpSolution.RbRampYes.Checked then
Ramprate:=true else RampRate:=false;
  cekiter:=1;
  repeat
    if (loadjam[cekiter]<pgenmin) or
(loadjam[cekiter]>pgenmax) then
    begin
      showmessage('DED not possible with scheduled generation
number '+inttostr(cekiter));
      prosesrun:=false;
      break;
    end;
    cekiter:=cekiter+1;
  until cekiter=numper+1;

  if cekiter=numper+1 then
  begin
    prosesrun:=true;
    getfilename(filename,d,f,e);
    title1 := 'File Name : '+f+'.'+e;
    assign(ff,'data.dum');
    rewrite(Ff);

    for i:=1 to numper do
    begin
    datadump(loadjam[i], unitsebelum,Ff) ;
     if solution_type = lamsearch then
lambda_search_dispatch( loadjam[i], lambda );
    if solution_type = enhanced then enhanced_lambda(
loadjam[i], lambda );
     {if solution_type = tbllookup  then
table_lookup_dispatch( lambda );}
    output_routine( loadjam[i], ff, lambda ) ;
    total_period[i]:=totalcost ;
      for k := 1 to ngen do
      begin
        unitsebelum[k]:=p[k];
        data_period[i,k]:=p[k];
      end;
      if ProsesRun = false then
      begin
        showmessage('DED not possible between scheduled
generation number '+inttostr(i-1)+' and '+inttostr(i));
        break;
      end;
    end;
```

```
    output_final(ff);

      for k := 1 to ngen do
      begin
        unitsebelum[k]:=0;
      end;
      close(fF);
      if ProsesRun = true then
      begin
        frmDEDSetupSolution.Close ;
      end;
    end;


 end;

procedure TFrmDEDSetUpSolution.Button1Click(Sender: TObject);
begin
 title1:='EDC FILE ' +fileName;

 proses;

end;


procedure TFrmDEDSetUpSolution.EdnumlChange(Sender: TObject);
var s:string;
n:integer;
k:integer;
begin
val(trim(ednuml.Text),n,k);
if n>24 then
begin
  n:= 24;

end;
if n<=1 then n:=1;
sgload.ColCount :=N+1;
for k:= 1 to N do
begin
  str(k,s);
  sgload.Cells[k,0]:='Periode '+trim(s);

end;
NUMPER:=N;
end;

procedure TFrmDEDSetUpSolution.SgLoadKeyUp(Sender: TObject;
var Key: Word;
  Shift: TShiftState);
```

```
var i,k:integer; s:string;
begin
for i := 1 to numper do
    begin
      s:=sgload.Cells[i,1];
      val(s,loadjam[i],k);
    end;

end;

procedure TFrmDEDSetUpSolution.FormClose(Sender: TObject;
  var Action: TCloseAction);
var i:integer;
begin
for i :=1 to ngen do
begin
pmax[i]:=unitmax[i];
pmin[i]:=unitmin[i];
end;
end;



procedure TFrmDEDSetUpSolution.rblamsearchClick(Sender:
TObject);
begin
  solution_type  := lamsearch;
end;

procedure TFrmDEDSetUpSolution.rbELIClick(Sender: TObject);
begin
  solution_type  := enhanced;
end;

end.
```

RIWAYAT PENULIS

Nadia Dwiatmo terlahir sebagai anak kedua dari dua bersaudara di Jakarta pada tanggal 27 Juni 1991. Penulis telah menyelesaikan pendidikan Diploma dan memperoleh gelar Ahli madya (A.Md) dari kampus Politeknik Negeri Bandung (POLBAN) pada tahun 2012 dengan bidang keahlian Teknik Listrik. Tahun 2014 sampai sekarang praktikan melanjutkan pendidikan untuk mendapatkan gelar Strata 1, melalui program Lintas Jalur di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember Surabaya dengan program studi Teknik Sistem Tenaga.

Email: nadia2atmo@gmail.com

---Halaman Ini Sengaja Dikosongkan---